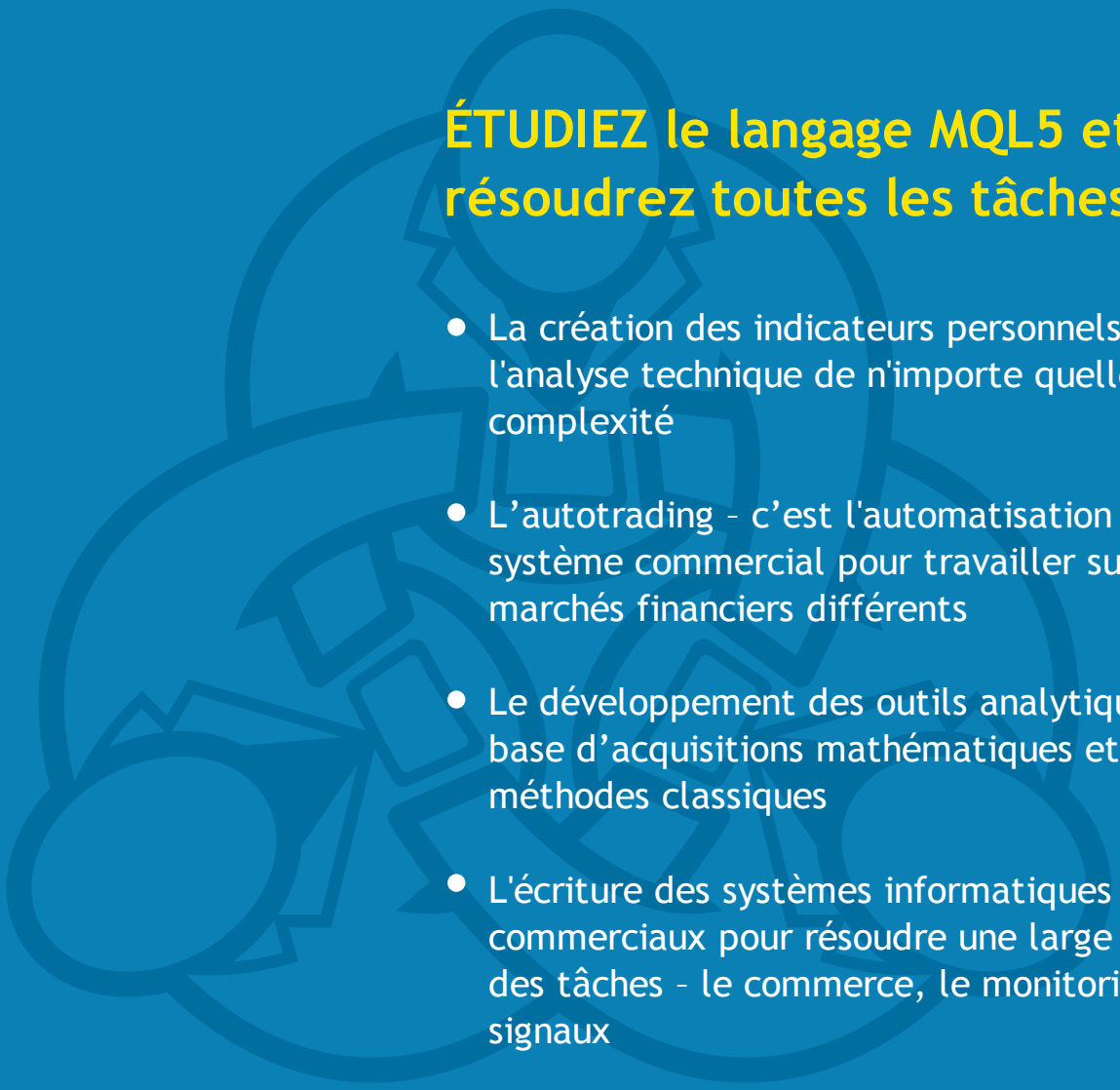
Le manuel de référence du langage METAQUOTES LANGUAGE 5

pour le terminal de client MetaTrader 5



ÉTUDIEZ le langage MQL5 et résoudrez toutes les tâches:

- La création des indicateurs personnels de l'analyse technique de n'importe quelle complexité
- L'autotrading – c'est l'automatisation du système commercial pour travailler sur les marchés financiers différents
- Le développement des outils analytiques à la base d'acquisitions mathématiques et des méthodes classiques
- L'écriture des systèmes informatiques commerciaux pour résoudre une large gamme des tâches – le commerce, le monitoring, des signaux

© 2000-2015, MetaQuotes Software Corp.

Teneur

Référence MQL5 51

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

Références MQL5

MetaQuotes Language 5 (MQL5) - c'est un langage intégré de la programmation des stratégies commerciales, qui a été mis au point par la compagnie MetaQuotes Software Corp. en basant sur son expérience pluriannuel de création des plateaux commerciales et informatiques. Ce langage permet de créer ses propres programmes-experts (Expert Advisors), qui automatisent la gestion des processus commerciales et qui sont parfaitement adapté à la mise en oeuvre des ses propres stratégies commerciales. En outre dans MQL5 on peut créer ses propres indicateurs techniques (Custom Indicators), scripts (Scripts) et des librairies des fonctions (Libraries).

La structure MQL5 comprend un grand nombre de fonctions nécessaires pour l'analyse des cotations présentes et antérieures à venir, les indicateurs et les fonctions principales de gestion des positions commerciales et de contrôle se sont integrés dans ce programme

Pour écrire le code du programme on utilise un éditeur expert textuel MetaEditor 5 qui extrait par la couleur des différentes constructions du langage MQL5, qui permet à l'utilisateur de mieux naviguer dans le texte du système expert. On utilise MetaQuotes Language Dictionary comme le système d'aide dans le langage MQL5.

Guide de référence contient des fonctions réparties dans des catégories des fonctions, des opérations, des mots réservés et les autres constructions du langage et permet de trouver la description de chaque élément qu'on utilise dans le langage.

Les programmes écrits en MetaQuotes Language 5, ont des caractéristiques et des buts différents:

- **Conseiller-expert** — c'est un système méchanique commercial (SMC), qui est attaché au graphique déterminé. Le Conseiller-expert est exécuté lorsqu'un événement se produit, il peut gérer l'événement, les événements de l'initialisation et déinitialisation, un événement d'arrivée de nouveau tick, un événement de minuterie, profondeur de l'évolution du marché d'événement, des événement de graphique et des événement des utilisateurs. Le Conseiller expert peut travailler à la fois au mode de l'information de la possibilité de passer un acte et passer des actes automatiquement sur le compte commercial en les envoyant directement sur le serveur de commerce. Les Conseillers-experts se sont stockés dans le répertoire *dossier_terminal\MQL5\Experts*.
- **Indicateur d'utilisateur** — indicateur technique écrit par l' utilisateur en addition aux indicateurs, qui sont déjà integrés au terminal de client. Les indicateurs d'utilisateur ainsi que les intégrés ne peuvent pas commercer automatiquement et ils sont destinés pour la réalisation des fonctions analytiques.
  Les indicateurs d' utilisateur se sont stockés dans le répertoire *dossier_terminal\MQL5\Indicators*
- **Script** — est un programme destiné à une seule exécution de certaines actions. A la différence des Conseillers-experts les scripts ne traitent pas aucuns événements sauf un événement du lancement (Cela nécessite la fonction de gestionnaire OnStart dans un script). Les scripts se sont stockés dans le répertoire *dossier_terminal\MQL5\Scripts*
- **Librairie** — une bibliothèque de fonctions personnalisées conçue pour le stockage et la distribution des blocs de programmes des utilisateurs utilisés le plus souvent. Les librairies ne peuvent pas être exécutées de manière indépendante. Les bibliothèques se sont stockés dans le répertoire *dossier_terminal\MQL5\Libraries*
- **Fichier inclus** — c' est un texte source des blocs fréquemment utilisés des programmes d' utilisateur. Tels fichiers peuvent être inclus dans les textes sources des conseillers-experts, les scripts, des indicateurs d'utilisateur, des bibliothèques lors de la compilation. L'utilisation des fichiers inclus est préférable que l'utilisation des bibliothèques en raison des charges

© 2000-2015, MetaQuotes Software Corp.

supplémentaires en appelant des fonctions des bibliothèques.
Les fichiers inclus peuvent être stockés dans le même répertoire que le fichier source, dans ce cas on utilise la directive #include avec les guillemets doubles. Un autre endroit pour stocker les fichiers inclus est dans le répertoire - *dossier_terminal\MQL5\Include*, dans ce cas on utilise la directive #include avec les chevrons.

© 2000-2015, MetaQuotes Software Corp.

Les principes du langage

Le langage MetaQuotes Language 5 (MQL5) est le langage orientée objet de programmation de haut niveau et destinée à la rédaction des stratégies automatiques commerciales, des indicateurs techniques d' utilisateurs pour l'analyse des différents marchés financiers. Elle permet non seulement d'écrire une variété de systèmes experts, conçus pour fonctionner en temps réel, mais aussi créer leurs propres outils graphiques pour aider à prendre des décisions commerciales.

MQL5 est basée sur la conception du langage de programmation C++ largement répandu, en comparaison de MQL4 le nouveau langage a maintenant des énumérations, des structures, des classes et le traitement des événements. En augmentant le nombre des types insérés principaux, l'interaction des programmes exécutés dans MQL5 avec d'autres applications par l'intermédiaire de dll est facilitée au maximum. La syntaxe MQL5 est similaire à la syntaxe du langage C + + et ça permet de lui transférer des programmes des langages de programmation moderne.

Pour vous aider à étudier le langage MQL5, tous les thèmes sont regroupés dans les catégories suivantes:

- Syntaxe
- Types de données
- Opérations et expressions
- Opérateurs
- Fonctions
- Variables
- Préprocesseur
- Programmation orientée objet

© 2000-2015, MetaQuotes Software Corp.

Syntaxe

Quant à la syntaxe, le langage de programmation des stratégies commerciales MQL5 ressemble au langage de programmation C++, sauf pour quelques caractéristiques:

- aucun arithmétique d'adresse
- aucun opérateur goto;
- l'énumération anonyme ne peut pas être déclarée;
- aucun héritage multiple

Voir aussi

les Enumérations, les Structures et les classes, l'Héritage

© 2000-2015, MetaQuotes Software Corp.

Commentaires

Les commentaires de multiligne se commencent par la paire des symboles /* et se terminent par la paire */. Ces commentaires ne peuvent pas être imbriqués. Les commentaires d'une seule ligne se commencent par la paire des symboles //, se terminent par le symbole de nouvelle ligne et peuvent être imbriqués dans les commentaires de multiligne. Les commentaires sont autorisés partout où les espaces sont acceptés, ils peuvent avoir n'importe quel nombre d'espaces.

Exemples:

```
//--- Commentaire d'une seule ligne
/*  Multi-
    ligne         // Commentaire imbriqué d'une seule ligne
    commentaire
*/
```

© 2000-2015, MetaQuotes Software Corp.

Identificateurs

Les identifiants sont utilisés comme des noms des variables et des fonctions. La longueur de l'identificateur ne peut pas dépasser 63 caractères.

Les caractères autorisés pour l'écriture de l'identificateur: les chiffres 0-9, les lettres majuscules et minuscules latins a-z et A-Z, reconnus comme des caractères différents, le caractère de soulignement (_). Le premier caractère ne peut pas être un chiffre.

L'identificateur ne doit pas coïncider avec mot réservé.

Exemples:

```
NAME1 name1 Total_5 Paper
```

Voir aussi

Variables, Fonctions

© 2000-2015, MetaQuotes Software Corp.

Mots réservés

Les identificateurs suivants sont enregistrés comme des mots réservés, dont chacun correspond à une certaine action, et ne peut pas être utilisé dans un autre sens:

Types de données

bool	enum	struct
char	float	uchar
class	int	uint
color	long	ulong
datetime	short	ushort
double	string	void

Spécificateurs d'accès

const	private	protected
public	virtual	

Classes de mémoire

extern	input	static

Opérateurs

break	do	operator
case	else	return
continue	for	sizeof
default	if	switch
delete	new	while

Autres

false	#define	#property
this	#import	template
true	#include	typename

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

Les types des donées

Tout programme opère avec les données. Les données peuvent être des types différents selon de leur destination. Par exemple, on utilise les données du type entier pour l'accès aux éléments du tableau. Les données de prix ont le type de l'exactitude double avec la virgule flottante. C'est rattaché au fait qu'aucun type spécial n'est pas prévu dans MQL5 pour les données de prix.

Les données des types différents se sont traitées avec la vitesse différente. Les données de nombre entier sont traités plus vite. Pour traiter les données de précision doubles un coprocesseur spécial est utilisé. Cependant à cause de la complexité de la représentation intérieure des données avec la virgule flottante elles sont traitées davantage que les données entières.

Les données de chaîne sont traitées le plus longtemps. Cela est dû à l'allocation et à la redistribution dynamique de la mémoire de l'ordinateur.

Les types de données fondamanteux sont:

- entiers (char, short, int, long, uchar, ushort, uint, ulong)
- logiques (bool)
- constantes littérales (ushort)
- les chaînes (string)
- avec la virgule flottante (double, float)
- la couleur (color)
- la date et les heures (datetime)
- les énumérations (enum)

Types de données complex sont:

- les structures;
- les classes.

Dans les termes de la POO les types complexes de données s'appellent les types abstraits de données.

Les types color et datetime ont le sens seulement pour le confort de la représentation et de l'entrée des paramètres, spécifiés du dehors - du tableau des propriétés du conseiller ou l'indicateur d'utilisateur (l'onglet "Inputs"). Les données des types color et datetime se présentent en forme des nombres entières. Les types entiers avec les types avec la virgule flottante s'appellent les types arithmétiques (numériques).

Dans les expressions on utilise la conformation implicite des types, si on n'indique pas la conformation évidente.

Voir aussi

Conformation des types

© 2000-2015, MetaQuotes Software Corp.

Les types entiers

Les types entiers sont représentés au langage MQL5 par onze espèces. Certains types peuvent être utilisés avec les autres, si c'est nécessaire par la logique du programme mais dans ce cas-là il est nécessaire de rappeler les règles de la conversion des types.

Le tableau ci-dessous donne les caractéristiques de chaque type. En outre, dans une dernière colonne le type correspondant dans le langage de programmation C++ est indiqué pour chaque type.

Type	Grandeur en bits	Valeur minimale	Valeur maximale	Analogue C++
char	1	-128	127	char
uchar	1	0	255	unsigned char, BYTE
bool	1	0(false)	1(true)	bool
short	2	-32 768	32 767	short, wchar_t
ushort	2	0	65 535	unsigned short, WORD
int	4	- 2 147 483 648	2 147 483 647	int
uint	4	0	4 294 967 295	unsigned int, DWORD
color	4	-1	16 777 215	int, COLORREF
long	8	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	__int64
ulong	8	0	18 446 744 073 709 551 615	unsigned __int64
datetime	8	0 (1970.01.01 0:00:00)	32 535 244 799 (3000.12.31 23:59:59)	__time64_t

On peut aussi présenter les valeurs des types entiers en forme des constantes numériques, les literals colorées, les literals de la date-temps, les constantes symboliques et les énumérations.

Voir aussi

La conversion des données, Les constantes des types numéraux

© 2000-2015, MetaQuotes Software Corp.

Types char, short, int et long

char

Le type char prend 1 byte de mémoire (8 bits) et permet d'exprimer dans le système de numération binaire 2^8 de valeur =256. Le type char peut contenir les valeurs tant positives que négatives. La gamme des valeurs est de -128 à 127.

uchar

Le type uchar prend aussi 1 byte de mémoire, comme le type char, mais à la différence de lui, uchar est destiné pour les valeurs positives seulement. La valeur minimale est zéro, la valeur maximale est 255. La première lettre u au nom du type d'uchar est l'abrègement pour le mot unsigned (non signé).

short

Le type short a la grandeur 2 bytes (16 bits) et il permet d'exprimer l'éventail des valeurs égales au 2 à l' échelon 16: 2^16=65 536. Comme le type short est un signe et contient à la fois les valeurs positives et négatives, la gamme de valeurs est comprise entre -32 768 et 32 767.

ushort

Le type non signé short est le type ushort, qui a également la grandeur 2 bytes. La valeur minimale est zéro, la valeur maximale est 65 535.

int

Le type int a la grandeur 4 bytes (32 bits). La valeur minimale est -2 147 483 648, la valeur maximale est 2 147 483 647.

uint

Le type non signé uint prend 4 byte de mémoire et permet d'exprimer les valeurs entiers de 0 à 4 294 967 295.

long

Le type long a la grandeur 8 bytes (64 bits). La valeur minimale est -9 223 372 036 854 775 808, la valeur maximale est 9 223 372 036 854 775 807.

ulong

Le type ulong prend aussi 8 bytes et permet de stocker les valeurs de 0 à 18 446 744 073 709 551 615.

Exemples:

```
char  ch=12;
short sh=-5000;
int   in=2445777;
```

© 2000-2015, MetaQuotes Software Corp.

Comme les types non signés ne sont pas destinés à conserver des valeurs négatives, la tentative de fixer une valeur négative peut conduire à des conséquences inattendues. Un tel script simple conduira à une boucle infinie:

```
//--- boucle infinie
void OnStart()
  {
   uchar  u_ch;

   for(char ch=-128;ch<128;ch++)
     {
      u_ch=ch;
      Print("ch = ",ch," u_ch = ",u_ch);
     }
  }
```

La variante correcte est:

```
//--- variante correcte
void OnStart()
  {
   uchar  u_ch;

   for(char ch=-128;ch<=127;ch++)
     {
      u_ch=ch;
      Print("ch = ",ch," u_ch = ",u_ch);
      if(ch==127) break;
     }
  }
```

Résultat:

```
   ch= -128  u_ch= 128
   ch= -127  u_ch= 129
   ch= -126  u_ch= 130
   ch= -125  u_ch= 131
   ch= -124  u_ch= 132
   ch= -123  u_ch= 133
   ch= -122  u_ch= 134
   ch= -121  u_ch= 135
   ch= -120  u_ch= 136
   ch= -119  u_ch= 137
   ch= -118  u_ch= 138
   ch= -117  u_ch= 139
   ch= -116  u_ch= 140
   ch= -115  u_ch= 141
   ch= -114  u_ch= 142
   ch= -113  u_ch= 143
   ch= -112  u_ch= 144
```

© 2000-2015, MetaQuotes Software Corp.

```
   ch= -111  u_ch= 145
    ...
```

Exemples:

```
//--- Il est interdit de consérver les valeurs négatives dans les types non-signés
uchar  u_ch=-120;
ushort u_sh=-5000;
uint   u_in=-401280;
```

Hexadécimals: les chiffres 0-9, les lettres a-f ou A-F pour les valeurs 10-15; commencent par 0x ou 0X.

Exemples:

```
0x0A, 0x12, 0X12, 0x2f, 0xA3, 0Xa3, 0X7C7
```

Для целочисленных переменных значения можно задавать в бинарном виде с помощью префикса B. Например, можно закодировать рабочие часы торговой сессии в переменную типа int и использовать информацию о них согласно требуемому алгоритму:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- для рабочих часов ставим 1, для нерабочих указываем 0
   int AsianSession   =B'111111111'; // азиатская сессия с 0:00 часов до 9:00
   int EuropeanSession=B'111111111000000000'; // европейская сессия  9:00 - 18:00
   int AmericanSession =B'111111110000000000000011'; // американская 16:00 - 02:00
//--- выведем числовые значения сессий
   PrintFormat("Asian session hours as value =%d",AsianSession);
   PrintFormat("European session hours as value is %d",EuropeanSession);
   PrintFormat("American session hours as value is %d",AmericanSession);
//--- а теперь выведем строковые представления рабочих часов сессий
   Print("Asian session ",GetHoursForSession(AsianSession));
   Print("European session ",GetHoursForSession(EuropeanSession));
   Print("American session ",GetHoursForSession(AmericanSession));
//---
  }
//+------------------------------------------------------------------+
//| возвращает рабочие часы сессии в строковом виде                  |
//+------------------------------------------------------------------+
string GetHoursForSession(int session)
  {
//--- для проверки используем битовые операции AND и сдвиг на один бит влево <<=1
//--- начинаем проверять с самого младшего бита
   int bit=1;
   string out="working hours: ";
//--- будем проверять все 24 бита начиная с нулевого до 23 включительно
```

© 2000-2015, MetaQuotes Software Corp.

```
   for(int i=0;i<24;i++)
     {
      //--- получим состояние бита bit в числе number
      bool workinghour=(session&bit)==bit;
      //--- добавим номер часа в сообщение
      if(workinghour )out=out+StringFormat("%d ",i);
      //--- сдвигаем на один бит влево для проверки значения следующего
      bit<<=1;
     }
//--- сформированная строка
   return out;
  }
```

Voir aussi

Conformation des types

© 2000-2015, MetaQuotes Software Corp.

Les constantes de caractères

Les symboles, comme l'élément de ligne dans MQL5 sont des index dans l'ensemble de caractères Unicode. Ils sont les valeurs de 16 bits, lesquelles on peut transformer en nombres entières et avec lesquelles on peut manipuler les opérations entières tels que l'addition et la soustraction.

N'importe quel symbole séparé conclu aux guillemets simples, ou le code hexadécimal ASCII du symbole dans l'aspect '\x10' est la constante symbolique et a le type ushort. Par exemple, l'inscription de l'aspect '0' présente d'elle-même la valeur numérique 30, correspondant à l'index, selon laquelle dans le tableau des symboles se trouve le symbole le zéro.

Exemple:

```
void OnStart()
  {
//---  définissons les constantes symboliques
   int symbol_0='0';
   int symbol_9=symbol_0+9; // recevons le symbole '9'
//--- déduisons les valeurs des constantes
   printf("Dans l'aspect décimal: symbol_0 = %d,  symbol_9 = %d",symbol_0,symbol_9);
   printf("Dans l'aspect hexadécimal: symbol_0 = 0x%x,  symbol_9 = 0x%x",symbol_0,symb
//--- établissons les constantes dans la chaîne
   string test="";
   StringSetCharacter(test,0,symbol_0);
   StringSetCharacter(test,1,symbol_9);
//--- Et voici comment ils apparaissent dans la chaîne
   Print(test);
  }
```

La barre oblique inversée c'est le symbole dirigeant pour le compilateur à l'analyse des chaînes constantes et les constantes symboliques dans le texte initial du programme. On peut présenter certains caractères, tels que les apostrophes ('), les guillemets ("), la barre oblique inversée (\), et les symboles dirigeants par la combinaison de symboles qui commencent par la barre oblique inversée(\),conformément à la table présentée ci-dessous :

Nom du symbole	Code mnémonique ou la représentation	Enregistrement dans MQL5	Valeur numérique
une nouvelle chaîne (la conversion de la chaîne)	LF	'\n'	10
la tabulation horizontale	HT	'\t'	9
le retour du chariot	CR	'\r'	13
la barre oblique inversée	\	'\\'	92
les apostrophes	'	'\''	39
les guillements	"	'\"'	34
le code hexadécimal	hhhh	'\xhhhh'	De 1 à 4 signes hexadécimaux

© 2000-2015, MetaQuotes Software Corp.

le code décimal	d	'\d'	le nombre décimal de 0 jusqu'à 65535

Le résultat n'est pas défini si le symbole qui se diffère des énumérées suivit la barre oblique inversée.

Exemple

```
void OnStart()
  {
//--- déclarons les constantes symboliques
   int a='A';
   int b='$';
   int c='©';       // code 0xA9
   int d='\xAE';    // code du symbole ®
//--- imprimons les constantes
   Print(a,b,c,d);
//--- ajoutons le symbole à la chaîne
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- remplaçons le symbole dans la chaîne
   StringSetCharacter(test,0,b);
   Print(test);
//--- remplaçons le symbole dans la chaîne
   StringSetCharacter(test,0,c);
   Print(test);
//--- remplaçons le symbole dans la chaîne
   StringSetCharacter(test,0,d);
   Print(test);
//--- présentons les symboles comme le nombre
   int a1=65;
   int b1=36;
   int c1=169;
   int d1=174;
//--- ajoutons le symbole à la chaîne
   StringSetCharacter(test,1,a1);
   Print(test);
//--- remplaçons le symbole dans la chaîne
   StringSetCharacter(test,1,b1);
   Print(test);
//--- remplaçons le symbole dans la chaîne
   StringSetCharacter(test,1,c1);
   Print(test);
//--- remplaçons le symbole dans la chaîne
   StringSetCharacter(test,1,d1);
   Print(test);
  }
```

Comme il a été déjà dit ci-dessus, la valeur de la constante symbolique (ou la variable) représente l'index dans le tableau des symboles, mais puisque l'index est la nombre entière, il est admissible de l'enregistrer par des chemins différents.

© 2000-2015, MetaQuotes Software Corp.

```
void OnStart()
  {
//---
   int a=0xAE;      // code du symbole ® correspond à la literal '\xAE'
   int b=0x24;      // code du symbole ® correspond à la literal '\x24'
   int c=0xA9;      // code du symbole ® correspond à la literal '\xA9'
   int d=0x263A;    // code du symbole ☺ correspond à la literal '\x263A'
//--- déduisons les valeurs
   Print(a,b,c,d);
//--- ajoutons le symbole à la chaîne
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- remplaçons le symbole dans la chaîne
   StringSetCharacter(test,0,b);
   Print(test);
//--- remplaçons le symbole dans la chaîne
   StringSetCharacter(test,0,c);
   Print(test);
//--- remplaçons le symbole dans la chaîne
   StringSetCharacter(test,0,d);
   Print(test);
//--- codes des couleurs
   int a1=0x2660;
   int b1=0x2661;
   int c1=0x2662;
   int d1=0x2663;
//--- ajoutons le symbole de la pique
   StringSetCharacter(test,1,a1);
   Print(test);
//--- ajoutons le symbole des coeurs
   StringSetCharacter(test,2,b1);
   Print(test);
//--- ajoutons le symbole des carreaux
   StringSetCharacter(test,3,c1);
   Print(test);
//--- ajoutons le symbole de trèfle
   StringSetCharacter(test,4,d1);
   Print(test);
//--- exemple des literals symboliques dans la chaîne
   test="Dame\x2660As\x2662";
   printf("%s",test);
  }
```

La représentation intérieure de la literal symbolique - le type ushort. Les constantes symboliques peuvent accepter les valeurs de 0 jusqu'à 65535.

Voir aussi

StringSetCharacter(), StringGetCharacter(), ShortToString(), ShortArrayToString(), StringToShortArray()

© 2000-2015, MetaQuotes Software Corp.

Le type datetime

Le type datetime est destiné pour le stockage de la date et de l'heure comme le nombre de secondes écoulées depuis le 01 janvier 1970. Il prend le 8 bytes de mémoire:

Les constantes de la date et de l' heure peuvent être présentés comme la ligne literal, qui se compose de 6 parties, représentant la valeur numérique de l' an, du mois, les nombres (ou le nombre, le mois, l' année), l' heure, la minute et la seconde. La constante est encadrée par les guillements simples et commence par le symbole D.

La gamme des significations est du 1 janvier 1970 jusqu'au 31 decembre 3000. La date (l' année, le mois, le nombre) ou le temps (les heures, les minutes, les secondes) ou tous ensemble peuvent être omis.

Il est désirable d'indiquer l'année, le mois et le jour à la tâche, autrement le compilateur donnera l'avertissement sur l'enregistrement incomplet littéral.

Exemples:

```
datetime NY=D'2015.01.01 00:00';     // le temps de l'arrivée de l'an 2015
datetime d1=D'1980.07.19 12:30:27';  // l'année le mois le jour les heures les minutes
datetime d2=D'19.07.1980 12:30:27';  // est équivalent au D'1980.07.19 12:30:27';
datetime d3=D'19.07.1980 12';        // est équivalent au D'1980.07.19 12:00:00'
datetime d4=D'01.01.2004';           // est équivalent au D'01.01.2004 00:00:00'
datetime compilation_date=__DATE__;             // la date de la compilation
datetime compilation_date_time=__DATETIME__;    // la date et le temps de la compilati
datetime compilation_time=__DATETIME__-__DATE__;// le temps de la compilation
//--- les exemples des déclarations, sur lesquelles on reçoit les avertissements du co
datetime warning1=D'12:30:27';       // est équivalent au D'[la date de la compilation
datetime warning2=D'';               // est équivalent au __DATETIME__
```

Voir aussi

Structure de la date, Date et temps, TimeToString, StringToTime

© 2000-2015, MetaQuotes Software Corp.

Type color

Le type color est destiné à conserver l' information du couleur et prend 4 bytes de la mémoire. Le premier byte est ignoré, les autres 3 bytes contiennent les composants RGB.

Les constantes de couleur peut être représentés de trois façons: littéralement, en nombres entiers ou à l' aide du nom (pour les Web-couleurs seulement).

La représentation literale se compose de trois parties, qui représente les valeurs numériques de l'intensité de trois principales composantes de la couleur: rouge (red), vert (green), bleu (blue). La constante commence par le symbole C et est entourée de guillemets simples. Les valeurs numériques de l'intensité de composante de la couleur sont dans la gamme de 0 à 255.

La représentation entière est écrite dans une forme d'un nombre hexadécimal ou d'un nombre décimal. Le nombre hexadécimal a la forme 0x00BBGGRR, où RR - c'est la valeur de l'intensité de composante rouge du couleur, GG - de composante verte, et BB - de composante bleue. Les constantes décimales ne sont pas directement reflétées dans RGB. Ils représentent une valeur décimale de la représentation entière hexadécimal.

Les couleurs spécifiques reflètent ce qu'on appelle le Web-couleurs définies.

Exemples:

```
//--- littéraux
C'128,128,128'    // gris
C'0x00,0x00,0xFF' // bleu
//les noms de couleur
clrRed               // rouge
clrYellow            // jaune
clrBlack             // noir
//--- représentations entières
0xFFFFFF          // blanc
16777215          // blanc
0x008000          // vert
32768             // vert
```

Voir aussi

Ensemble des couleurs Web, ColorToString, StringToColor, Conformation des types

© 2000-2015, MetaQuotes Software Corp.

Le type bool

Le type bool est destiné à conserver les valeurs logiques true (la vérité) ou false (le faux), dont la représentation numérique est 1 ou 0, proportionnellement.

Exemples:

```
bool a = true;
bool b = false;
bool c = 1;
```

La représentation intérieure - est un nombre entier qui a la grandeur 1 byte. Il est nécessaire de noter qu'on peut utiliser les types entiers ou matériels ou les expressions de ces types au lieu de type bool dans les expressions logiques, le compilateur ne produira pas d' erreur. Dans ce cas, la valeur zéro sera interprétée comme fausse et toutes les autres valeurs - comme vraies.

Exemples:

```
   int i=5;
   double d=-2.5;
   if(i) Print("i = ",i,"et a la valeur true");
   else Print("i = ",i,"et a la valeur false");

   if(d) Print("d = ",d,"et a la valeur true");
   else Print("d = ",d,"et a la valeur false");

   i=0;
   if(i) Print("i = ",i,"et a la valeur true");
   else Print("i = ",i,"et a la valeur false");

   d=0.0;
   if(d) Print("d = ",d,"et a la valeur true");
   else Print("d = ",d,"et a la valeur false");

//---  résultats d'exécution
//   i= 5 et a la valeur true
//   d= -2.5 et a la valeur true
//   i= 0 et a la valeur false
//   d= 0 et a la valeur false
```

Voir aussi

Opérations logiques, Les priorités et l'ordre des opérations

© 2000-2015, MetaQuotes Software Corp.

Les énumérations

Des données du type enum appartiennent à un ensemble limité des certaines données. La définition du type énumeré:

```
enum nom du_type_énumeré
  {
   liste_des valeurs
  };
```

La liste des valeurs d'énumération est une liste de variables, séparées par des virgules.

Exemple:

```
enum months  // énumération des constantes nommeés
   {
    January,
    February,
    March,
    April,
    May,
    June,
    July,
    August,
    September,
    October,
    November,
    December
   };
```

Après que l'énumération est déclarée, un nouveau type de données de 4 bytes évalué de nombre entier apparaît. La déclaration d'un nouveau type des données permet au compilateur de contrôler strictement les types des paramètres transmis parce que l'énumération introduit de nouvelles constantes nommées. Dans l'exemple ci-dessus, la constante nommée January a la valeur 0, February a la valeur 1, December a la valeur 11.

La règle: si une certaine valeur n' est pas attribuée à la constante nommée qui est le membre de l'énumération alors ça valeur sera générée automatiquement. Si elle est le premier membre de l'énumération, la valeur 0 sera attribuée. Pour tous les membres séquentiels la valeur sera calculée sur la base de la valeur des membres antérieurs en y ajoutant l'unité.

Exemple:

```
enum intervals  // énumération des constantes nommées
   {
    month=1,     // intervalle d'un mois
    two_months,  // deux mois
    quarter,     // trois mois - quartier
    halfyear=6,  // semestre
    year=12,     // an - 12 mois
   };
```

© 2000-2015, MetaQuotes Software Corp.

Notes

- A la différence de C++, la valeur de la représentation intérieure du type énumeré dans MQL5 est toujours égale à 4 bytes. C'est-à-dire, sizeof(months) renvoie la valeur 4.
- A la différence de C++ l'énumération anonyme ne peut pas être déclarée dans MQL5. C'est-à-dire, un nom unique doit toujours être spécifié après le mot-clé enum.

Voir aussi

Conformation des types

© 2000-2015, MetaQuotes Software Corp.

Les types réels (double, float)

Les types réels (ou les types à virgule flottante) représentent des valeurs ayant une partie fractionnaire. Dans le langage MQL5 il existe deux types pour les nombres à virgule flottante. La méthode pour la représentation de nombres réels dans la mémoire informatique est définie par la norme IEEE 754 et elle est indépendante des plates-formes, des systèmes opérationnels ou des langages de programmation.

Type	Grandeur en bytes	Valeur minimale positive	Valeur maximale	Exactitude de représentation	Analogue C++
float	4	1.175494351 e-38	3.402823466e +38	7chiffres significatifs	float
double	8	2.225073858 5072014e-308	1.7976931348 623158e+308	15chiffres significatifs	double

Le nom double signifie, que l' exactitude de ces nombres dépasse le double celle de nombres de type float. Dans la plupart des cas le type double est plus commode. Dans la plupart des cas l' exactitude limitée de nombre float n' est pas suffisante. La raison pour laquelle le type float est encore utilisé - c' est l' économie de la mémoire de stockage (c' est important pour les gros tableaux de nombres réels).

Les constantes à virgule flottante se composent d'une partie entière, du point (.) et d'une partie fractionnaire. Les parties entières et fractionnaires se présentent des séquences de chiffres décimaux.

Exemples:

```
double a=12.111;
double b=-956.1007;
float  c =0.0001;
float  d =16;
```

Il y a une façon scientifique d' écrire des constantes réels, souvent cette méthode d'enregistrement est plus compacte que la traditionnelle

Exemple:

```
double c1=1.12123515e-25;
double c2=0.000000000000000000000000112123515; // 24 zéros après la virgule décimal

Print("1. c1 =",DoubleToString(c1,16));
// Résultat: 1. c1 = 0.0000000000000000

Print("2. c1 =",DoubleToString(c1,-16));
// Résultat: 2. c1 = 1.1212351499999999e-025

Print("3. c2 =",DoubleToString(c2,-16));
// Résultat: 3. c2 = 1.1212351499999999e-025
```

© 2000-2015, MetaQuotes Software Corp.

Il faut se rappeler que les nombres réels sont stockés dans la mémoire de l'ordinateur avec une certaine exactitude limitée dans le système de numération binaire, pendant que le système de numération décimale est plus convenu. C'est pourquoi beaucoup de nombres qui sont justement enregistrés dans le système décimal peut être rédigé uniquement en tant que la fraction infini dans le système binaire.

Par exemple, le nombre de 0.3 et 0.7 se sont représentés dans l'ordinateur par les fractions infinies, tandis que le nombre de 0,25 est stocké exactement, car il représente la puissance de deux.

A ce propos il est recommandé catégoriquement de ne pas comparer les deux nombres réels pour l'égalité, car une telle comparaison n'est pas correcte.

Exemple:

```
void OnStart()
  {
//---
   double three=3.0;
   double x,y,z;
   x=1/three;
   y=4/three;
   z=5/three;
   if(x+y==z) Print("1/3 + 4/3 == 5/3");
   else Print("1/3 + 4/3 != 5/3");
// Résultat: 1/3 + 4/3 != 5/3
  }
```

S'il faut comparer deux nombres réels pour l'égalité, on peut faire ça par deux façons différentes. La première méthode consiste à comparer la différence entre deux chiffres avec un petit nombre qui spécifie l'exactitude de la comparaison.

Exemple:

```
bool EqualDoubles(double d1,double d2,double epsilon)
  {
   if(epsilon<0) epsilon=-epsilon;
//---
   if(d1-d2>epsilon) return false;
   if(d1-d2<-epsilon) return false;
//---
   return true;
  }
void OnStart()
  {
   double d_val=0.7;
   float  f_val=0.7;
   if(EqualDoubles(d_val,f_val,0.000000000000001)) Print(d_val,"equals ",f_val);
   else Print("Different: d_val = ",DoubleToString(d_val,16),
              "  f_val = ",DoubleToString(f_val,16));
// Résultat: Different: d_val = 0.7000000000000000   f_val = 0.6999999880790710
  }
```

© 2000-2015, MetaQuotes Software Corp.

Il faut noter que la valeur de l'epsilon dans l'exemple ci-dessus ne peut être moins de la constante prédéterminé DBL_EPSILON. La valeur de cette constante est 2.2204460492503131e-016. Pour le type float la constante relevante est FLT_EPSILON = 1.192092896e-07. La signification de ces valeurs est telle que c'est la plus petite valeur, qui satisfait à la condition 1.0+DBL_EPSILON != 1.0 (pour les nombres du type float 1.0+FLT_EPSILON != 1.0).

La deuxième méthode consiste à comparer la différence normalisée de deux nombres réels avec le zéro. Il est inutile de comparer la différence de nombres normalisés avec un zéro parce que toute opération mathématique avec des nombres normalisés donne un résultat non- normalisé.

Exemple:

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0) return(true);
   else return(false);
  }
void OnStart()
  {
   double d_val=0.3;
   float  f_val=0.3;
   if(CompareDoubles(d_val,f_val)) Print(d_val,"equals ",f_val);
   else Print("Different: d_val = ",DoubleToString(d_val,16),
              "  f_val = ",DoubleToString(f_val,16));
// Résultat: Different: d_val = 0.3000000000000000   f_val = 0.3000000119209290
  }
```

Quelques opérations du coprocesseur mathématiques peuvent donner le nombre réel invalide qui ne peut pas être utilisé dans les opérations mathématiques et dans les opérations des comparaisons parce que le résultat d'opérations avec les nombres réels invalides n' est pas défini. Par exemple, lorsqu'on essaye de calculer l'arc sinus de 2, le résultat sera l'infinité négative.

Exemple:

```
   double abnormal = MathArcsin(2.0);
   Print("MathArcsin(2.0) =",abnormal);
// Résultat:  MathArcsin(2.0) = -1.#IND
```

Sauf l'infinité négative il y a l'infinité positive et PuN (pas un nombre). Pour déterminer que ce nombre n'est pas valide, on peut utiliser la fonction MathIsValidNumber(). Selon le standard IEEE ils ont la représentation spéciale de machine. Par exemple, l'infinité positive pour le type double a la représentation en bits 0x7FF0 0000 0000 0000.

Exemples:

```
struct str1
  {
   double d;
  };
struct str2
  {
   long l;
```

© 2000-2015, MetaQuotes Software Corp.

```
  };

//--- commencons
   str1 s1;
   str2 s2;
//---
   s1.d=MathArcsin(2.0);         // dériver un nombre invalide -1.#IND
   s2=s1;
   printf("1.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFFF000000000000;      // nombre invalide -1.#QNAN
   s1=s2;
   printf("2.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF7000000000000;      // pas un nombre le plus grand SNaN
   s1=s2;
   printf("3.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF8000000000000;      // pas un nombre le plus petit QNaN
   s1=s2;
   printf("4.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FFF000000000000;      // pas un nombre le plus grand QNaN
   s1=s2;
   printf("5.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF0000000000000;      // infinité positive 1.#INF et pas un nombre le plus p
   s1=s2;
   printf("6.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFF0000000000000;      // infinité négative -1.#INF
   s1=s2;
   printf("7.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x8000000000000000;      // zéro négatif -0.0
   s1=s2;
   printf("8.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FE0000000000000;      // 0.5
   s1=s2;
   printf("9.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FF0000000000000;      // 1.0
   s1=s2;
   printf("10.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FEFFFFFFFFFFFFF;      // le plus grand nombre normalisé (MAX_DBL)
   s1=s2;
   printf("11.  %.16e %I64X",s1.d,s2.l);
```

© 2000-2015, MetaQuotes Software Corp.

```
//---
   s2.l=0x0010000000000000;     // le plus petit positif normalisé (MIN_DBL)
   s1=s2;
   printf("12.  %.16e %.16I64X",s1.d,s2.l);
//---
   s1.d=0.7;                    // montrons que le nombre 0.7 - est une fraction infi
   s2=s1;
   printf("13.  %.16e %.16I64X",s1.d,s2.l);
/*
1.  -1.#IND00 FFF8000000000000
2.  -1.#QNAN0 FFFF000000000000
3.   1.#SNAN0 7FF7000000000000
4.   1.#QNAN0 7FF8000000000000
5.   1.#QNAN0 7FFF000000000000
6.   1.#INF00 7FF0000000000000
7.  -1.#INF00 FFF0000000000000
8.  -0.000000 8000000000000000
9.   0.500000 3FE0000000000000
10.  1.000000 3FF0000000000000
11.  1.7976931348623157e+308 7FEFFFFFFFFFFFFF
12.  2.2250738585072014e-308 0010000000000000
13.  6.9999999999999996e-001 3FE6666666666666
*/
```

Aussi à voir

DoubleToString, NormalizeDouble, Constantes des types numériques

© 2000-2015, MetaQuotes Software Corp.

Le type string

Le type string est destiné pour stocker les chaînes de texte. Le texte de chaîne est une séquence de caractères au format Unicode avec le zéro final à la fin. A la variable string peut être attribué la constante de chaîne. La constante chaîne est une séquence des caractères Unicode, entourée de guillemets doubles: "C'est est une constante chaîne".

S'il est necéssaire d'introduire dans la chaîne les guillemets doubles ("), il faut mettre avant la barre oblique inversée (\). N'importe quel caractère spécial des constantes peut être introduit dans la chaîne, s'il y a la barre oblique inversée avant (\).

Exemples:

```
string svar="This is a character string";
string svar2=StringSubstr(svar,0,4);
Print("Symbole de copyright\t\x00A9");
FileWrite(handle,"cette chaîne contient le symbole de conversion de chaîne\n");
string MT5path="C:\\Program Files\\MetaTrader 5";
```

On peut décomposer les longues chaînes constantes en plusieurs parties pour le confort de la lecture du code initial sans opération de l'addition. Ces parties se réuniront automatiquement à une longue chaîne pendant la compilation:

```
//--- déclarons une longue chaîne constante
   string HTML_head="<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\"'
                    " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n"
                    "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n"
                    "<head>\n"
                    "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=ut
                    "<title>Trade Operations Report</title>\n"
                    "</head>";
//--- déduisons une chaîne constante dans un journal
   Print(HTML_head);
  }
```

Voir aussi

Conformation des types, Fonctions de chaîne, FileOpen, FileReadString, FileWriteString

© 2000-2015, MetaQuotes Software Corp.

Les structures et les classes

Les structures

La structure est un ensemble d'éléments du type arbitraire (sauf le type void). Ainsi, la structure regroupe logiquement les données relatives de types différents.

La déclaration de la structure

Le type structuré de données est déterminé par la description suivante:

```
struct nom_du structure
  {
   description_des éléments
  };
```

Le nom de la structure ne peut pas être utilisé comme un identificateur (nom de la variable ou de la fonction). Il faut noter que les éléments de la structure dans MQL5 se suivent directement sans alignement. Dans le langage C++ un tel ordre est fait au compilateur à l'aide de l'instruction.

```
#pragma pack(1)
```

S'il faut faire un autre alignement dans la structure, il faut utiliser les membres auxiliaires, -" le remplisseur" des grandeurs justes.

Exemples:

```
struct trade_settings
  {
   uchar  slippage;     // valeur du glissement admissible - est 1 byte
   char   reserved1;    // 1 byte de l'espace
   short  reserved2;    // 2 bytes de l'espace
   int    reserved4;    // encore 4 bytes de l'espace. On a garanti l'alignement de la
   double take;         // valeur du prix et de fixation du profit
   double stop;         // la valeur du prix du Stop-protecteur
  };
```

Une telle description des structures alignées est necéssaire seulement pour le transfert aux fonctions dll importées.

Attention: cet exemple illustre des données incorrectement conçues. Ce serait mieux d'abord de déclarer les données take et stop de la plus grande valeur du type double, et déclarez ensuite le membre slippage du type uchar. Dans ce cas-là, la représentation intérieure de données sera toujours la même sans tenir compte de la valeur spécifiée dans *#pragma pack()*.

Si la structure contient des variables du type string et/ou l'objet du tableau dynamique, le compilateur assigne à une telle structure un implicite constructeur, qui remit à zéro tous les membres de structure du type string et l'initialisation correcte pour l'objet d'un tableau dynamique.

Les structures simples

Les structures qui ne contiennent pas des chaînes et des objets de tableaux dynamiques sont appelés

© 2000-2015, MetaQuotes Software Corp.

les structures simples; les variables des telles structures peuvent être librement copiées l'un à l'autre, même s'ils sont de différentes structures. On peut transmettre les variables des structures simples, ainsi que leurs tableaux à titre des paramètres aux fonctions importées de DLL.

L'accès aux membres de la structure

Le nom de la structure est un nouveau type de données et permet de déclarer des variables de ce type. La structure peut être déclarée qu'une seule fois au sein du projet. L'accès aux membres des structures effectue à l'aide de l' opération point (.).

Exemple:

```
struct trade_settings
  {
   double take;         // valeur du prix de la fixation du profit
   double stop;         // valeur du prix du Stop - protecteur
   uchar  slippage;     // valeur du glissement admissible
  };
//--- on a créé et initialisé une variable de type trade_settings
trade_settings my_set={0.0,0.0,5};
if (input_TP>0) my_set.take=input_TP;
```

Les classes

Les classes se diffèrent des structures dans le suivant:

- on utilise le mot-clé class dans les déclarations;
- par défaut, tous les membres de la classe ont le spécificateur de l' accès private s'il n'y a pas une autre indication. Les membres-les données de la structure ont le type d' accès public, s'il n'y a pas une autre indication;
- les objets de la classe ont toujours un tableau des fonctions virtuelles, même si aucune fonction virtuelle n'est pas déclarée dans la classe. Les structures ne peuvent pas avoir des fonctions virtuelles;
- On peut appliquer l'opérateurnewaux objets de la classe, on ne peut pas appliquer cet opérateur aux structures;
- les classes peuvent être héritées seulement par les classes, les structures peuvent être héritées seulement par les structures.

Les classes et les structures peuvent avoir un constructeur explicite et le destructeur. Dans le cas si le constructeur est défini, l'initialisation de variable de type de la structure ou la classe à l'aide de la séquence initialisée est impossible.

Exemple:

```
struct trade_settings
  {
   double take;         // valeur du prix de la fixation du profit
   double stop;         // valeur du prix du Stop - protecteur
   uchar  slippage;     // valeur du glissement admissible
   //--- constructeur
          trade_settings() { take=0.0; stop=0.0; slippage=5; }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- destructeur
         ~trade_settings() { Print("C'est la fin"); }
  };
//--- Le compilateur produira un message d'erreur que l'initialisation est impossible
trade_settings my_set={0.0,0.0,5};
```

Les constructeurs et les destructeurs

Le constructeur - c'est une fonction spéciale qui est appelée automatiquement lorsque vous créez une structure ou une classe et elle est utilisé d'habitude pourl'initialisationdes membres de la classe. Ensuite nous parlerons seulement des classes, et cela se rapporte aux structures, si on ne stipule pas l'autre Le nom du constructeur doit correspondre au nom de la classe. Le constructeur n'a pas le type rendu (on peut spécifier le typevoid).

Les membres définis de la classe - les chaînes, les tableaux dynamiques et les objets, demandant l'initialisation en tout cas seront initialisés, indépendamment de la présence du constructeur.

Chaque classe peut avoir quelques constructeurs, qui se différent par le nombre de paramètres et les listes d'initialisation. Le constructeur qui nécessite l'indication des paramètres, s'appelle le concepteur paramétrique.

Le constructeur sans paramètres s'appelle**le constructeur par défaut**. Si aucun constructeur n'est pas déclaré dans une classe, le compilateur créera lui-même le constructeur par défaut pendant la compilation.

```
//+------------------------------------------------------------------+
//| La classe pour le travail avec la date                           |
//+------------------------------------------------------------------+
class MyDateClass
  {
private:
   int               m_year;          // l'année
   int               m_month;         // le mois
   int               m_day;           // le jour du mois
   int               m_hour;          // une heure dans vingt-quatre heures
   int               m_minute;        // les minutes
   int               m_second;        // les secondes
public:
   //--- le constructeur par défaut
                     MyDateClass(void);
   //--- le constructeur avec les paramètres
                     MyDateClass(int h,int m,int s);
  };
```

On peut déclarer le constructeur dans la description de la classe, et puis définir son corps. Par exemple, voilà comment peuvent être définis deux constructeur de la classe MyDateClass:

```
//+------------------------------------------------------------------+
//| Le constructeur par défaut                                       |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
MyDateClass::MyDateClass(void)
  {
//---
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=mdt.hour;
   m_minute=mdt.min;
   m_second=mdt.sec;
   Print(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Le constructeur avec les paramètres                              |
//+------------------------------------------------------------------+
MyDateClass::MyDateClass(int h,int m,int s)
  {
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=h;
   m_minute=m;
   m_second=s;
   Print(__FUNCTION__);
  }
```

On rémplit tous les membres de la classe dans le constructeur par défaut à l'aide de la fonction TimeCurrent(), dans le constructeur avec les paramètres on remplit seulement les valeurs de l'heure. Les autres membres de la classe (m_year, m_month et m_day) seront initialisés automatiquement par la date courante.

Le constructeur par défaut a la destination spéciale pendant l'initialisation du tableau des objets de sa classe. Le constructeur dont tous les paramètres ont les valeurs par défaut **n'est pas** le constructeur par défaut. Montrons cela à l'exemple:

```
//+------------------------------------------------------------------+
//| La classe avec le constructeur par défaut                        |
//+------------------------------------------------------------------+
class CFoo
  {
   datetime          m_call_time;     // le temps du dernier appel à l'objet
public:
   //--- le constructeur avec le paramètre ayant l'importance par défaut, n'est pas le
                     CFoo(const datetime t=0){m_call_time=t;};
   //--- le constructeur du copiage
                     CFoo(const CFoo &foo){m_call_time=foo.m_call_time;};
```

© 2000-2015, MetaQuotes Software Corp.

```
   string ToString(){return(TimeToString(m_call_time,TIME_DATE|TIME_SECONDS));};
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
// CFoo foo; // on ne peut pas utiliser une telle variante - le constructeur par défau
//--- les variantes admissibles de la création de l'objet CFoo
   CFoo foo1(TimeCurrent());     // l'appel évident du constructeur paramétrique
   CFoo foo2();                  // l'appel évident du constructeur paramétrique avec
   CFoo foo3=D'2009.09.09';      // l'appel implicite du constructeur paramétrique
   CFoo foo40(foo1);             // l'appel évident du constructeur du copiage
   CFoo foo41=foo1;              // l'appel implicite du constructeur du copiage
   CFoo foo5;                    // l'appel évident du constructeur par défaut (si le
                                 // le constructeur paramétrique avec le paramètre par
//--- les variantes admissibles de la réception des pointeurs CFoo
   CFoo *pfoo6=new CFoo();       // la création d'un objet dynamique et la réception d
   CFoo *pfoo7=new CFoo(TimeCurrent());// encore une variante de la création dynamique
   CFoo *pfoo8=GetPointer(foo1); // maintenant pfoo8 indique à l'objet foo1
   CFoo *pfoo9=pfoo7;            // pfoo9 et pfoo7 indiquent au même objet
   // CFoo foo_array[3];         // on ne peut pas utiliser une telle variante - le co
//--- déduisons les valeurs m_call_time
   Print("foo1.m_call_time=",foo1.ToString());
   Print("foo2.m_call_time=",foo2.ToString());
   Print("foo3.m_call_time=",foo3.ToString());
   Print("foo4.m_call_time=",foo4.ToString());
   Print("foo5.m_call_time=",foo5.ToString());
   Print("pfoo6.m_call_time=",pfoo6.ToString());
   Print("pfoo7.m_call_time=",pfoo7.ToString());
   Print("pfoo8.m_call_time=",pfoo8.ToString());
   Print("pfoo9.m_call_time=",pfoo9.ToString());
//--- supprimons les objets dynamiquement créés
   delete pfoo6;
   delete pfoo7;
   //delete pfoo8;  // évidemment il ne faut pas supprimer pfoo8, car il indique à l'o
   //delete pfoo9;  // évidemment il ne faut pas supprimer pfoo9, car il indique au mê
  }
```

Si on commentera les chaînes dans cet exemple

```
   //CFoo foo_array[3];     // on ne peut pas utiliser une telle variante - le construc
```

ou

```
   //CFoo foo_dyn_array[];  // on ne peut pas utiliser une telle variante - le construc
```

le compilateur donnera l'erreur sur eux "default constructor is not defined".

Si le constructeur déclaré par l'utilisateur a la classe, le constructeur par défaut ne sera pas généré par le compilateur. Cela signifie que si le constructeur avec les paramètres est déclaré dans la classe, mais

© 2000-2015, MetaQuotes Software Corp.

le constructeur par défaut n'est pas déclaré, on ne peut pas déclarer les tableaux des objets de cette classe. Voici sur quel script le compilateur déclarera l'erreur:

```
//+------------------------------------------------------------------+
//| La classe sans constructeur par défaut                           |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
                     CFoo(string name) { m_name=name;}
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---recevons l'erreur pendant la compilation "default constructor is not defined"
   CFoo badFoo[5];
  }
```

Dans l'exemple donné la classe CFoo a le constructeur paramétrique déclaré - dans tels cas le compilateur pendant la compilation ne crée pas automatiquement le constructeur par défaut. En même temps à la déclaration du tableau des objets il est supposé que tous les objets doivent êtrecréés et initialisés automatiquement. Il faut appeler le constructeur par défaut lors de l'initialisation automatique d'un objet, mais depuis le constructeur par défaut n'est pas explicitement déclaré n'est pas généré automatiquement par le compilateur, la création d'un tel objet est impossible. Pour cette raison le compilateur génère l'erreur au moment de la compilation.

Il y a une syntaxe spéciale pour l'initialisation de l'objet à l'aide du constructeur. Les initialiseurs du constructeur (les constructions spéciales pour l'initialisation) pour les membres de la structure ou la classe peuvent être spécialisées dans la liste de l'initialisation.

La liste d'initialisation - c'est une liste des initialiseurs, séparés par les virgules, qui est suivi après deux points par la liste des paramètres du constructeur et précèdele corps(qui est avant l'accolade ouvrante). Il y a quelques exigences:

- on peut utiliser les listes de l'initialisation seulement dansles constructeurs;
- dans la liste de l'initialisation on ne peut pas initialiserles membres des parents;
- après la liste de l'initialisation doit aller la définition (la réalisation) de la fonction.

Montrons l'exemple de plusieurs constructeurs pour initialiser les membres de la classe.

```
//+------------------------------------------------------------------+
//| La classe pour stocker le nom et le prénom du personnage         |
//+------------------------------------------------------------------+
class CPerson
  {
   string            m_first_name;     // le prénom
   string            m_second_name;    // le nom
public:
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- le constructeur par défaut vide
                     CPerson() {Print(__FUNCTION__);};
   //--- le constructeur paramétrique
                     CPerson(string full_name);
   //---le constructeur avec la liste d'initialisation
                     CPerson(string surname,string name): m_second_name(surname), m_fi
   void PrintName(){PrintFormat("Name=%s Surname=%s",m_first_name,m_second_name);};
  };
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
CPerson::CPerson(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>=0)
     {
      m_first_name=StringSubstr(full_name,0,pos);
      m_second_name=StringSubstr(full_name,pos+1);
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- recevons l'erreur "default constructor is not defined"
   CPerson people[5];
   CPerson Tom="Tom Sawyer";                       // Tom Sawyer
   CPerson Huck("Huckleberry","Finn");             // Huckleberry Finn
   CPerson *Pooh = new CPerson("Whinnie","Pooh");  // Whinnie Pooh
   //--- déduisons les valeurs
   Tom.PrintName();
   Huck.PrintName();
   Pooh.PrintName();

   //---supprimons l'objet dynamiquement créé
   delete Pooh;
  }
```

Dans ce cas la classe CPerson a trois constructeurs:

1. évident le constructeur par défaut, qui permet de créer le tableau des objets de la classe donnée;
2. le constructeur avec un paramètre, qui prend un nom complet à titre du paramètre et le divise en nom et le prénom par un espace découvert;
3. le constructeur avec deux paramètres, qui contient la liste d'initialisation. Les initialiseurs - m_second_name(surname) et m_first_name(name).

Remarquez comment l'initialisation est remplacée l'affectation à l'aide de la liste. Les membres séparés doivent être initialisés comme:

© 2000-2015, MetaQuotes Software Corp.

```
Le membre_de la classe (la liste des expressions)
```

Dans la liste de l'initialisation les membres peuvent aller en n'importe quel ordre, mais à tout cela les membres de la classe seront initialisés selon l'ordre de leur déclaration. Cela signifie que dans le troisième constructeur tout d'abord sera initialisé le membre m_first_name, puisqu'il est déclaré le premier, et seulement après lui sera initialisé le membre m_second_name. Ceci devrait être considéré dans le cas quand l'initialisation de certains membres de la classe dépend des valeurs des autres membres de la classe.

Si dans la classe de base on n'a pas déclaré le constructeur par défaut et de plus un ou quelques constructeurs avec les paramètres sont déclarés, il faut absolument appeler un des constructeurs de la classe de base dans la liste de l'initialisation. Il est séparé par la virgule comme les membres ordinaires de la liste et sera appelé en premier lieu pendant l'initialisation de l'objet indépendamment de la situation dans la liste de l'initialisation.

```
//+------------------------------------------------------------------+
//| La classe de base                                                |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
   //---le constructeur avec la liste d'initialisation
                     CFoo(string name) : m_name(name) { Print(m_name);}
  };
//+------------------------------------------------------------------+
//| Le descendant de la classe CFoo                                  |
//+------------------------------------------------------------------+
class CBar : CFoo
  {
   CFoo              m_member;      //le membre de la classe est l'objet du prédécesse
public:
   //--- le constructeur par défaut dans la liste de l'initialisation appelle le const
                     CBar(): m_member(_Symbol), CFoo("CBAR") {Print(__FUNCTION__);}
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   CBar bar;
  }
```

Dans cet exemple à la création de l'objet "bar" le constructeur CBar() sera appelé par défaut, où tout d'abord on appelle le constructeur pour le prédécesseur CFoo, et après le constructeur pour le membre m_member.

Le destructeur - est une fonction spéciale qui est appelée automatiquement quand on détruit l'objet de la classe. Le nom du destructeur est écrit comme un nom de classe avec un tilde (~). Les chaînes, les tableaux dynamiques et les objets, exigeant la déinitialisation en tout cas seront déinitialisés indépendamment de la présence de destructeur. S' il y a le destructeur, ces actions seront réalisées

© 2000-2015, MetaQuotes Software Corp.

après l'appel de destructeur.

Les destructeurs sont toujours virtuels, même s'ils se sont declarés avec le mot-clé virtual ou non.

La définition des méthodes des classes

Les fonctions-méthodes de la classe peuvent être définies comme à l'intérieur de la classe et en dehors de la déclaration de la classe. Si la méthode est définie dans la classe, son corps suivit après la déclaration de la méthode.

Exemple:

```
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
   int               GetType()                  { return(m_type);        }
   void              Left()                     { m_xpos-=SHAPE_SIZE;    }
   void              Right()                    { m_xpos+=SHAPE_SIZE;    }
   void              Rotate()                   { m_prev_turn=m_turn; if(++m_turn>3) m
   virtual void      Draw()                     { return;                }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
  };
```

Les fonctions de SetRightBorder(int border) au Draw() sont déclarées et définies à l' intérieur de la classe CTetrisShape.

Le constructeur CTetrisShape() et les méthodes CheckDown(int& pad_array[]), CheckLeft(int& side_row[]) et CheckRight(int& side_row[]) sont déclarées seulement à l' intérieur de la classe, mais ne sont pas définies. Les définitions de ces fonctions seront suivits selon le code. Pour définir la méthode à l'extérieur de la classe, on utilise l'opération de résolution de contexte, le nom de classe est utilisé comme le contexte.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Constructeur de la classe fondamentale                           |
//+------------------------------------------------------------------+
void CTetrisShape::CTetrisShape()
  {
   m_type=0;
   m_ypos=0;
   m_xpos=0;
   m_xsize=SHAPE_SIZE;
   m_ysize=SHAPE_SIZE;
   m_prev_turn=0;
   m_turn=0;
   m_right_border=0;
  }
//+------------------------------------------------------------------------+
//| Le contrôle de la capacité de bouger en bas (pour le bâton et le cube) |
//+------------------------------------------------------------------------+
bool CTetrisShape::CheckDown(int& pad_array[])
  {
   int i,xsize=m_xsize/SHAPE_SIZE;
//---
   for(i=0; i<xsize; i++)
     {
      if(m_ypos+m_ysize>=pad_array[i]) return(false);
     }
//---
   return(true);
  }
```

Les modificateurs d'accès public, protected et private

En développant une nouvelle classe, il est recommandé de restreindre l'accès aux membres de l'extérieur. Pour ces buts, on utilise les mots clés private ou protected. Dans ce cas-là l'accès aux données cachées peut être effectué seulement des fonctions- les méthodes de la même classe. Si le mot clé protected est utilisé, alors l'accès aux données cachées peut être effectué des méthodes des classes - les successeurs de cette classe. La même méthode peut être utilisée pour restreindre l'accès aux fonctions-les méthodes de la classe.

Si on a besoin d' ouvrir un accès aux membres et/ou les méthodes de la classe, on utilise le mot clé public.

Exemple:

```
class CTetrisField
  {
private:
   int               m_score;                          // compte
   int               m_ypos;                           // position actuelle de figu
   int               m_field[FIELD_HEIGHT][FIELD_WIDTH]; // matrice de DOM
   int               m_rows[FIELD_HEIGHT];             // numération des rangs de DO
   int               m_last_row;                       // dernier rang libre
   CTetrisShape     *m_shape;                          // figure de tetris
   bool              m_bover;                          // jeu est fini
public:
   void              CTetrisField() { m_shape=NULL; m_bover=false; }
```

© 2000-2015, MetaQuotes Software Corp.

```
   void              Init();
   void              Deinit();
   void              Down();
   void              Left();
   void              Right();
   void              Rotate();
   void              Drop();
private:
   void              NewShape();
   void              CheckAndDeleteRows();
   void              LabelOver();
  };
```

Tout les membres et les méthodes de classe qui sont déclarés après le spécificateur **public:** (et jusqu'au prochain spécificateur d'accès), sont disponibles dans n'importe quelle référence du programme pour l'objet de cette classe. Dans cet exemple, ce sont les membres suivants: les fonctions CTetrisField(), Init(), Deinit(), Down(), Left(), Right(), Rotate() et Drop().

Tous les membres de la classe qui sont déclarés après le spécificateur d'accès aux éléments **private:** (et jusqu'au prochain spécificateur d'accès), sont disponibles seulement aux fonctions- les membres de cette classe. Les spécificateurs d'accès aux éléments toujours se terminent par le deux-points (:) et peuvent apparaître dans la définition de la classe plusieurs fois.

L'accès aux membres de la classe de base peut être redéfinie en termes d'héritage dans les classes dérivées.

Voir aussi

Programmation orientée objet

© 2000-2015, MetaQuotes Software Corp.

L'objet du tableau dynamique

Les tableaux dynamiques

Il est admis la déclaration pas plus que le tableau quadridimensionnel. Lorsqu'on déclare un tableau dynamique (le tableau de la valeur non-spécifiée dans la première paire de chevrons), le compilateur crée automatiquement la variable de la structure susnommée (l'objet du tableau dynamique) et fournit un code à l'initialisation correcte.

Les tableaux dynamiques se libèrent automatiquement à la sortie au-delà du domaine de la visibilité du bloc, où ils sont déclarés.

Exemple:

```
double matrix[][10][20]; // tableau dynamique tridimensionnel
ArrayResize(matrix,5);   // on a mis la grandeur de la première dimension
```

Les tableaux statiques

Quand toutes les dimensions de tableau significative sont explicitement spécifiées, le compilateur affecte préventivement la grandeur nécessaire de mémoire. Un tel tableau s'appelle statique. Néanmoins, le compilateur affecte la mémoire supplémentaire pour l'objet du tableau dynamique, qui (l'objet) est associée avec le tampon statique préalablement distribué (la partie de mémoire pour stocker le tableau).

Création d'un objet du tableau dynamique est due à la nécessité éventuelle de redonner ce tableau statique comme un paramètre dans une fonction quelconque.

Exemples:

```
double stat_array[5]; // tableau statique unidimensionnel
some_function(stat_array);
...
bool some_function(double& array[])
  {
   if(ArrayResize(array,100)<0) return(false);
   ...
   return(true);
  }
```

Des tableaux dans les structures

Quand un tableau statique est déclarée comme un membre d'une structure, un objet de tableau dynamique ne créé pas. C'est fait pour assurer la compatibilité des structures de données utilisé dans Windows API.

Pourtant, les tableaux statiques qui sont déclarés comme les membres des structures peuvent aussi être passées aux fonctions MQL5. Dans ce cas à la transmission du paramètre un objet temporaire du tableau dynamique lié au tableau statique - le membre de la structure sera crée.

© 2000-2015, MetaQuotes Software Corp.

Voir aussi

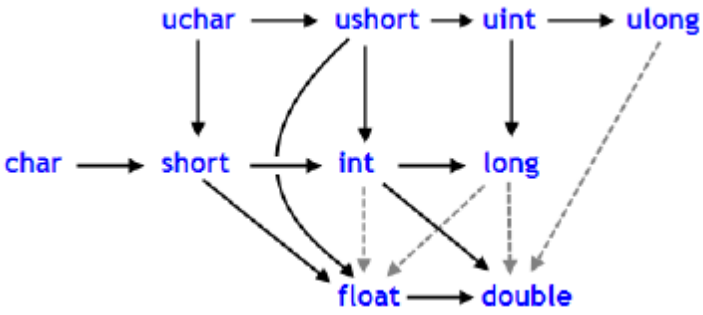
Opérations avec les tableaux, Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

La conformation des types

La conversion des types numériques

Souvent il est nécessaire de convertir un type numérique à une autre. Pas chaque type numérique peut être transformé en autre. Le schéma montre les transformations admissibles dans MQL5:



Les lignes continues avec des flèches indiquent les changements qui sont réalisées sans perte d'information. Au lieu du type char le type bool peut être utilisé (tous les deux prennent 1 byte de la mémoire), au lieu du type int on peut utiliser le type color (4 bytes), au lieu du type long il est admis le type datetime (tous les deux prennent 8 bytes). Les quatre lignes interrompus grises aussi marquées de flèches signifient les transformations, dans laquel la perte de l'exactitude peut être produite. Par exemple, le nombre de chiffres dans le nombre entier 123456789 (int) dépasse le nombre de chiffres qui peut être représenté par le type float.

```
   int n=123456789;
   float f=n;     // contenue f est égal à 1.234567892E8
   Print("n = ",n,"   f = ",f);
   // résultat n= 123456789    f= 123456792.00000
```

Le nombre converti dans le type float a le même ordre, mais l' exactitude un peu moins précise. Les transformations, inverses aux flèches noires, peuvent être exécutées avec la perte possible de données. Les transformations entre char et uchar, short et ushort, int et uint, long et ulong (les transformations aux deux côtés sont prévues), peuvent amener à la perte de l'information.

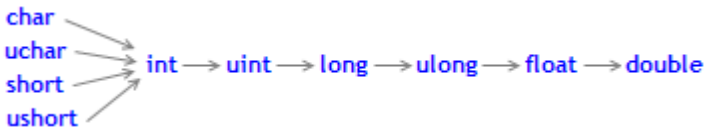
A la suite de la transformation de valeur de virgule flottante au type entier la partie fractionnaire est toujours rejetée. S'il faut arrondir le nombre avec la virgule flottante au plus proche nombre entier (ce que est plus utile dans beaucoup de cas), il faut utiliser la fonction MathRound().

Exemple:

```
//--- accélération de la pesanteur
   double g=9.8;
   double round_g=(int)g;
   double math_round_g=MathRound(g);
   Print("round_g = ",round_g);
   Print("math_round_g = ",math_round_g);

//   Résultat:
//   round_g = 9
//   math_round_g = 10
```

© 2000-2015, MetaQuotes Software Corp.

Si les deux valeurs se sont combinées par un opérateur binaire, avant l'exécution d'opération l'opérande d'un type plus bas est converti au plus haut type conformément à la priorité indiqué dans le schéma:



Les types des données char, uchar, short et ushort dans les opérations se sont rapportés absolument au type int.

Exemples:

```
   char   c1=3;
//--- premier exemple
   double d2=c1/2+0.3;
   Print("c1/2+0.3 =",d2);
// Résultat:   c1/2+0.3 = 1.3

//--- deuxième exemple
   d2=c1/2.0+0.3;
   Print("c1/2.0+0.3 =",d2);
// Résultat:   c1/2.0+0.3 = 1.8
```

L'expression calculée se compose de deux opérations. Dans le premier exemple, la variable c1 du type char est convertie à une variable temporaire du type int, que le second opérande dans l'opération de division, la constante 2, a le type int qui est plus haut. A la suite de la division de nombre entier 3/2 on reçoit la valeur 1, qui a le type int.

Dans la deuxième opération du premier exemple le second opérande est la constante 0.3, qui a le type double, donc le résultat de la première opération est convertie en variable temporaire du type double avec la valeur 1.0.

Dans le deuxième exemple la constante c1 du type char est convertie en variable temporaire du type double, que le second opérande dans l'opération de division, la constante 2.0, a le type double; les autres changements ne sont pas effectués.

La conformation des types numériques

Dans les expressions de langage MQL5 on peut utilisé la conformation explicite et implicite des types. La conformation explicite des types s'enregistre de la façon suivante:

```
var_1 = (type)var_2;
```

L'expression ou le résultat d'une exécution de fonction peut être comme une variable var_2. L'enregistrement fonctionnel de conformation explicite des types est possible:

```
var_1 = type(var_2);
```

Examinons une conformation explicite sur la base du premier exemple.

© 2000-2015, MetaQuotes Software Corp.

```
//--- troisième exemple
   double d2=(double)c1/2+0.3;
   Print("(double)c1/2+0.3 =",d2);
// Résultat:   (double)c1/2+0.3 = 1.80000000
```

Avant l'exécution d'opération de la division la variable c1 amène au type double. Maintenant, la constante entière 2 s' amène à la valeur 2.0 du type double, donc à la suite de la conversion un premier opérande a reçu le type double. En fait, la conversion explicite du type est une opération unaire.

En outre lorsqu'on essaye de faire la conformation des types le résultat peut aller au-delà de la gamme admissible. Dans ce cas, la troncature se produit. Par exemple:

```
   char c;
   uchar u;
   c=400;
   u=400;
   Print("c=",c); // résultat c=-112
   Print("u=",u); // résultat u=144
```

Avant de l'exécution d'opération (sauf l'opération d'affectation) il est converti en type qui a la plus grande priorité, et avant les opération d'affectation - en type entier.

Exemples:

```
   int    i=1/2;         // il n'y a pas de conformation des types, le résultat: 0
   Print("i=1/2  ",i);

   int k=1/2.0;          // expression est induit au type double,
   Print("k=1/2  ",k);   // après induit au type entier int, le résultat: 0

   double d=1.0/2.0;     // il n'y a pas de conformation des types, le résultat: 0.5
   Print("d=1/2.0; ",d);

   double e=1/2.0;       // expression est induit au type double,
   Print("e=1/2.0; ",e);// qui coïncide avec le type entier, le résultat: 0.5

   double x=1/2;         // expression de type int est induit au type entier double,
   Print("x=1/2; ",x);   // résultat: 0.0
```

Lorsque vous convertissez le type long/ulong au double peut passer la perte de l'exactitude: si l'entier est plus grand 9223372036854774784 ou moins de -9223372036854774784.

```
void OnStart()
  {
   long l_max=LONG_MAX;
   long l_min=LONG_MIN+1;
//--- trouvons un nombre entier maximum qui ne perd pas l'exactitude à la réduction ve
   while(l_max!=long((double)l_max))
      l_max--;
//--- trouvons un nombre entier minimale qui ne perd pas l'exactitude à la réduction v
   while(l_min!=long((double)l_min))
      l_min++;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- maintenant déduisons l'intervalle trouvé pour les nombres entières
   PrintFormat("A la réduction de la nombre entière vers double il doit "
               "être dans l'intervalle [%I64d, %I64d]",l_min,l_max);
//---Maintenant, nous allons voir ce qui se passe si le nombre est hors de cet interva
   PrintFormat("l_max+1=%I64d, double(l_max+1)=%.f, ulong(double(l_max+1))=%I64d",
               l_max+1,double(l_max+1),long(double(l_max+1)));
   PrintFormat("l_min-1=%I64d, double(l_min-1)=%.f, ulong(double(l_min-1))=%I64d",
               l_min-1,double(l_min-1),long(double(l_min-1)));
//--- recevons une telle conclusion
// A la réduction de la nombre entière vers double il doit être dans l'intervalle [-92
// l_max+1=9223372036854774785, double(l_max+1)=9223372036854774800, ulong(double(l_ma
// l_min-1=-9223372036854774785, double(l_min-1)=-9223372036854774800, ulong(double(l_
  }
```

La conformation pour le type string

Le type string a la plus haute priorité parmi les types simples. Donc, si un des opérande dans une opération est a le type string, un autre opérande sera jeté au type string automatiquement. Il faut noter que pour le type string une unique opération d'addition dyadique est admissible. La conformation explicite d'une variable du type string au type numérique est permise.

Exemples:

```
   string s1=1.0/8;            // expression est induit au type double,
   Print("s1=1.0/8; ",s1);     // après au type entier string,
// résultat:"0.12500000"(la chaîne qui contient 10 caractères)

   string s2=NULL;             // déinitialisation de la chaîne
   Print("s2=NULL; ",s2);      // résultat: une chaîne vide
   string s3="Ticket N"+12345; // expression est induit au type string
   Print("s=\"Ticket N"+12345,s3);

   string str1="true";
   string str2="0,255,0";
   string str3="2009.06.01";
   string str4="1.2345e2";
   Print(bool(str1));
   Print(color(str2));
   Print(datetime(str3));
   Print(double(str4));
```

La conformation des données des structures simples

On peut assigner les données de type des structures simples l'un à l'autre seulement dans le cas, si tous les membres de deux structures ont des types numériques. Ainsi dans l'opération d'attribution tous les deux opérandes, et à gauche et à droite, doivent être les types de structures. La conformation terme à terme ne se produit pas, un simple copiage se produit. Si les structures sont de différentes

© 2000-2015, MetaQuotes Software Corp.

valeurs, puis on copie le nombre de bytes qui correspond à une valeur plus petite. Ainsi l'absence des unions (union) est compensée dans le langage MQL5.

Exemples:

```
struct str1
  {
   double  d;
  };
//---
struct str2
  {
   long    l;
  };
//---
struct str3
  {
   int     low_part;
   int     high_part;
  };
//+------------------------------------------------------------------+
void OnStart()
  {
   str1 s1;
   str2 s2;
   str3 s3;
//---
   s1.d=MathArcsin(2.0);        // recevons un nombre invalide -1.#IND
   s2=s1;
   printf("1.  %f    %I64X",s1.d,s2.l);
//---
   s3=s2;
   printf("2.  high part of long %.8X   low part of long %.8X",
          s3.high_part,s3.low_part);
  }
```

Encore un exemple montre comment on peut organiser sa propre fonction pour obtenir du type color les représentation des couleurs en RGB (Red,Green,Blue). Pour faire ça créez deux structures de la même grandeur, mais avec un différent contenu intérieur. Pour plus de commodité, ajoutons la fonction au structure, qui rend la couleur à la répresentation de RGB comme une chaîne.

```
#property script_show_inputs
input color          testColor=clrBlue;// spécifier la couleur pour le test
//--- structure pour la répresentation en RGB
struct RGB
  {
   uchar             blue;          // composante bleue de couleur
   uchar             green;         // composante verte de couleur
   uchar             red;           // composante rouge de couleur
   uchar             empty;         // ce byte ne s' utilise pas
```

© 2000-2015, MetaQuotes Software Corp.

```
   string            toString();    // fonction pour recevoir la chaîne
  };
//--- fonction pour afficher les couleurs comme la chaîne
string RGB::toString(void)
  {
   string out="("+(string)red+":"+(string)green+":"+(string)blue+")";
   return out;
  }
//--- structure pour le stockage de type intégré color
struct builtColor
  {
   color             c;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- variable pour le stockage dans RGB
   RGB colorRGB;
//--- variable pour le stockage du type color
   builtColor test;
   test.c=testColor;
//--- conformation de deux structures en copiant le contenu
   colorRGB=test;
   Print("color ",test.c,"=",colorRGB.toString());
//---
  }
```

La conformation des types des indicateurs des classes de base aux indicateurs des classes dérivées

Les objets de la classe ouverte générée peuvent également être considérés comme des objets de la classe de base correspondante. Cela sert à quelques conséquences intéressantes. Par exemple, malgré le fait que les objets de différentes classes générées par une classe de base, peuvent se différer de façon significative l'un de l'autre, nous pouvons créer une liste reliée (List), parce que nous les considérons comme des objets du type de base. Mais le contraire n'est pas vrai: les objets de classe de base ne sont pas automatiquement les objets de la classe dérivée.

On peut utiliser la conformation explicite des types pour convertir les indicateurs de la classe de base aux indicateurs de la classe dérivée. Mais il faut être assuré en admissibilité d'une telle transformation, parce qu'autrement une erreur critique du temps d'exécution sera produite et le programme mql5 sera arrêté.

Voir aussi

Types des données

© 2000-2015, MetaQuotes Software Corp.

Le type void et la constante NULL

Syntaxiquement le type void est un type fondamental ainsi que les types char, uchar, bool, short, ushort, int, uint, color, long, ulong, datetime, float, double et string. Ce type est utilisé pour indiquer que la fonction ne rend pas de valeur, ou comme un paramètre de fonction il dénote l'absence de paramètres.

La variable constante prédéterminée **NULL** a le type *void*. Elle peut être allouée aux variables d'autres types fondamentaux sans conversion. La comparaison des variables du type fondamental avec la valeur **NULL** est autorisée**.**

Exemple:

```
//--- Si la chaîne n'est pas initialisée, allouez-y notre valeur prédéterminée
if(some_string==NULL) some_string="empty";
```

Aussi **NULL** peut être comparé avec les indicateurs aux objets, créés à l'aide de l' opérateur new.

Voir aussi

Variables, Fonctions

© 2000-2015, MetaQuotes Software Corp.

Les indicateurs des objets

En MQL5 il y a la possibilité de créer dynamiquement des objets du type complexe. C'est fait à l'aide de l'opérateur new, qui rend le descripteur de l'objet créé. Le descripteur a la valeur 8 bytes. Syntaxiquement, les descripteurs d'objet dans MQL5 sont semblables aux indicateurs dans C++.

Exemples:

```
MyObject* hobject= new MyObject();
```

In contrast to C++, the hobject variable from example above is not a pointer to memory, but rather an object descriptor. Furthermore, in MQL5 all objects in function parameters must be passed by reference. Below are examples of passing objects as function parameters:

```
class Foo
  {
public:
   string            m_name;
   int               m_id;
   static int        s_counter;
   //--- constructors and desctructors
                     Foo(void){Setup("noname");};
                     Foo(string name){Setup(name);};
                    ~Foo(void){};
   //--- initializes object of type Foo
   void              Setup(string name)
     {
      m_name=name;
      s_counter++;
      m_id=s_counter;
     }
  };
int Foo::s_counter=0;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declare an object as variable with its automatic creation
   Foo foo1;
//--- variant of passing an object by reference
   PrintObject(foo1);

//--- declare a pointer to an object and create it using the 'new' operator
   Foo *foo2=new Foo("foo2");
//--- variant of passing a pointer to an object by reference
   PrintObject(foo2); // pointer to an object is converted automatically by compiler

//--- declare an array of objects of type Foo
   Foo foo_objects[5];
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- variant of passing an array of objects
   PrintObjectsArray(foo_objects); // separate function for passing an array of object

//--- declare an array of pointers to objects of type Foo
   Foo *foo_pointers[5];
   for(int i=0;i<5;i++)
     {
      foo_pointers[i]=new Foo("foo_pointer");
     }
//--- variant of passing an array of pointers
   PrintPointersArray(foo_pointers); // separate function for passing an array of poin

//--- it is obligatory to delete objects created as pointers before termination
   delete(foo2);
//--- delete array of pointers
   int size=ArraySize(foo_pointers);
   for(int i=0;i<5;i++)
      delete(foo_pointers[i]);
//---
  }
//+------------------------------------------------------------------+
//| Objects are always passed by reference                           |
//+------------------------------------------------------------------+
void PrintObject(Foo &object)
  {
   Print(__FUNCTION__,": ",object.m_id," Object name=",object.m_name);
  }
//+------------------------------------------------------------------+
//| Passing an array of objects                                      |
//+------------------------------------------------------------------+
void PrintObjectsArray(Foo &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
     {
      PrintObject(objects[i]);
     }
  }
//+------------------------------------------------------------------+
//| Passing an array of pointers to object                           |
//+------------------------------------------------------------------+
void PrintPointersArray(Foo* &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
     {
      PrintObject(objects[i]);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
```

Voir aussi

Variables, Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

Les références. Le modificateur & et le mot-clé this

La transmission des paramètres selon la référence

A MQL5 on peut transmettre les paramètres des types simples selon la valeur ainsi que selon la référence, pendant que les paramètres des types complexes sont transmis toujours selon la référence. Pour l'indication au compilateur sur la nécessité de la transmission du paramètre selon la référence, devant le nom du paramètre on met le signe ET commercial **&**.

La transmission du paramètre selon la référence signifie la transmission de l'adresse de la variable, c'est pourquoi tous les changements produits sur le paramètre transmis selon la référence, se refléteront tout de suite dans la variable initiale aussi. En utilisant la transmission des paramètres selon la référence on peut organiser le retour simultanément de quelques résultats de la fonction. Pour prévenir le changement du paramètre transmis selon la référence, il est nécessaire d'utiliser le modificateur const.

Ainsi, si le paramètre d'entrée de la fonction est le tableau, l'objet de la structure ou de la classe, dans le titre de la fonction après le type de la variable et devant son nom on met le symbole '&'.

Exemple

```
class CDemoClass
  {
private:
   double            m_array[];

public:
   void              setArray(double &array[]);
  };
//+------------------------------------------------------------------+
//| le remplissage du tableau                                        |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
     ArrayResize(m_array,ArraySize(array));
     ArrayCopy(m_array, array);
     }
  }
```

Dans l'exemple cité ci-dessus on a déclaré la classe CDemoClass, qui contient le membre particulier - le tableau m_array[] du type double. La fonction setArray() a été déclarée, auquelle on transmit le tableau array[] selon la référence. Si on écrit le titre de la fonction sans indication de la transmission selon la référence, c'est-à-dire enlever le signe ET commercial, à la tentative de la compilation d'un tel code on donnera le message d'erreur.

Malgré le fait que le tableau est transmis selon la référence, nous ne pouvons pas faire l'attribution d'un tableau à l'autre. Il est nécessaire de faire le copiage de chaque élément du contenu de tableau-source au tableau- récepteur. La présence du symbole & pour les tableaux et les structures à la transmission à titre du paramètre de la fonction est obligatoire à la description de la fonction.

Le mot-clé this

© 2000-2015, MetaQuotes Software Corp.

La variable du type de la classe (l'objet) peut être transmise selon la référence, ainsi que selon l'indicateur. L'indicateur comme la référence sert pour recevoir l'accès à l'objet. Après la déclaration de l'index de l'objet il est nécessaire de lui appliquer l'opérateur new pour sa création et l'initialisation.

Le mot réservé **this** est destiné à la réception de la référence de l'objet sur lui-même, accessible à l'intérieur des méthodes de la classe ou la structure. **this** se réfère toujours à l'objet, dont la méthode il utilise, et l'expression GetPointer(this) donne l'indicateur de l'objet, dont le membre est la fonction, dans laquelle l'appel de fonction GetPointer() est effectué. A MQL5 les fonctions ne peuvent pas rendre les objets, mais il est permis de rendre l'indicateur de l'objet.

Ainsi, s'il est nécessaire que la fonction rendra l'objet, nous pouvons rendre l'indicateur de cet objet en forme de GetPointer(this). Ajoutons à la description de la classe CDemoClass la fonction getDemoClass(), qui rend l'indicateur de l'objet de cette classe.

```
class CDemoClass
  {
private:
   double            m_array[];

public:
   void              setArray(double &array[]);
   CDemoClass       *getDemoClass();
  };
//+------------------------------------------------------------------+
//| Le remplissage du tableau                                        |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
      ArrayResize(m_array,ArraySize(array));
      ArrayCopy(m_array,array);
     }
  }
//+------------------------------------------------------------------+
//| Rend l'indicateur personnel                                      |
//+------------------------------------------------------------------+
CDemoClass *CDemoClass::getDemoClass(void)
  {
   return(GetPointer(this));
  }
```

Les structures n'ont pas les indicateurs, on ne peut pas leur appliquer les opérateurs *new* et *delete*, et on ne peut pas utiliser GetPointer(this).

Voir aussi

Les indicateurs des objets, La création et la destruction des objets, Le Domaine de la visibilité et le temps de la vie des variables

© 2000-2015, MetaQuotes Software Corp.

Opérations et expressions

On attache une importance spéciale à quelques caractères et séquences de caractères. C'est soi-disant les symboles d'opérations, par exemple:

```
+ - * / %     symboles d'opérations arithmétiques
&& ||         symboles d'opérations logiques
= += *=       symboles d'opérations d'attribution
```

Les symboles d'opération sont utilisés dans les expressions et ont le sens quand les opérandes se sont appropriées. Aussi on accorde de l'importance aux signes de ponctuation. Les signes de ponctuation comprennent des parenthèses, des parenthèses figurées, une virgule, des deux-points et un point-virgule.

Les symboles des opérations, les signes de ponctuation et les espaces se servent pour séparer les éléments du langage.

Cette section contient la description des thèmes suivants:

- Expressions
- Opérations arithmétiques
- Opérations d'attribution
- Opérations des relations
- Opérations logiques
- Opérations bit à bit
- Autres opérations
- Priorités et l'ordre des opérations

© 2000-2015, MetaQuotes Software Corp.

Les expressions

L'expression se compose d'un ou de plusieurs opérandes et les symboles d'opérations. Une expression peut être écrite à plusieurs lignes.

Exemples:

```
a++; b = 10;          // plusieurs expressions sont situés sur une ligne
//---  expression est divisée en plusieurs lignes
x = (y * z) /
    (w + 2) + 127;
```

Une expression qui se termine par un point-virgule (;) est l'opérateur.

Voir aussi

Les priorités et l'ordre des opérations

© 2000-2015, MetaQuotes Software Corp.

Les opérations arithmétiques

Les opérations additives et multiplicatives appartiennent aux opérations arithmétiques:

```
Somme de variables                          i = j + 2;
Différence de variables                     i = j - 3;
Changement du signe                         x = - x;
Multiplication des valeurs                  z = 3 * x;
Quotient de la division                     i = j / 5;
Excès de la division                        minutes = time % 60;
Addition 1 à la valeur de la variable       i++;
Addition 1 à la valeur de la variable       ++i;
La soustraction 1 de la valeur de la variable  k--;
La soustraction 1 de la valeur de la variable  --k;
```

L'opération d'incrément et de décrément sont appliquées uniquement aux variables et ne s'appliquent pas aux constantes. L'incrément (++i) et le décrément (--k) préfixes s'appliquent à la variable juste avant l'utilisation de cette variable dans l'expression.

L'incrément (i++) et le décrément (k--) postfix s'appliquent à la variable juste avant l'utilisation de cette variable dans l'expression.

Une importante remarque

```
int i=5;
int k = i++ + ++i;
```

Les problèmes calculatoires peuvent apparaître au transfert de l'expression indiquée ci-dessus d'un milieu de la programmation à l'autre (par exemple, de Borland C ++ dans MQL5). En général, l'ordre des calculs dépend de la réalisation du compilateur. Il y a pratiquement deux moyens de la réalisation du postdécrément (postincrément):

1. le postdécrément (postincrément) est appliqué à la variable après le calcul de toute l'expression;
2. le postdécrément (postincrément) est appliqué à la variable immédiatement selon la place de l'opération.

Dans MQL5 à ce moment on a réalisé le premier moyen du calcul du postdécrément (postincrément). Mais même en possédant cette connaissance il vaut mieux ne pas expérimenter avec l'utilisation de la finesse donnée.

Exemples:

```
int a=3;
a++;             // expression valide
int b=(a++)*3;   // expression invalide
```

Voir aussi

Les priorités et l'ordre des opérations

© 2000-2015, MetaQuotes Software Corp.

Les opérations d'attribution

La valeur de l'expression qui inclut l'opération d'attribution, est la valeur d'opérande gauche après d'attribution:

```
Attribution de la valeur x à la variable y                    y = x;
```

Les opérations suivantes combinent les opérations arithmétiques ou les opérations bit à bit avec l'opération d'attribution:

```
Addition de la valeur de la variable y sur x                  y += x;
La soustraction de la valeur de la variable y sur x           y -= x;
Multiplication de la valeur de la variable y sur x            y *= x;
Division de la valeur de la variable y sur x                  y /= x;
Excès de la division de la valeur de la variable y sur x      y %= x;
Décalage de la représentation binaire de y à droite par x bits  y >>= x;
Décalage de la représentation binaire de y à gauche par x bits  y <<= x;
Opération bit à bit ET des représentations binaires y et x    y &= x;
Opération bit à bit OU des représentations binaires y et x    y |= x;
Opération bit à bit excluant OU
          des représentations binaires y et x                 y ^= x;
```

Les opérations bit à bit se produisent seulement avec des nombres entiers. A l'exécution de l'opération le progrès logique de la représentation y à droite/à gauche sur bit xest utilisé les 5 catégories cadettes binaires de la valeur x, les catégories principales sont rejetées, c'est-à-dire le progrès est produit sur bit 0-31.

En exécutant l'opération %= (la valeur y sur le module x) le signe du résultat coïncide avec le signe du dividende.

L'opérateur d'attribution peut être utilisé plusieurs fois dans une expression. Dans ce cas-là le traitement de l'expression est exécuté de droite à gauche:

```
y=x=3;
```

D'abord, la valeur 3 sera attribuée à la variable x, après la valeur de la variable x sera attribuée à la valeur y c 'est- à- dire 3 aussi.

Voir aussi

Les priorités et l'ordre des opérations

© 2000-2015, MetaQuotes Software Corp.

Les opérations des relations

La valeur logique FAUX est représentée par la valeur entière de zéro, et la valeur VRAI est représentée par n'importe quelle valeur non-zéro.

FAUX (0) ou VRAI (1) sont la valeur d'expressions qui contiennent les opérations des relations ou les opérations logiques.

```
Vrai, si a est égal à b                 a == b;
Vrai, si a n'est pas égal à b           a != b;
Vrai, si a est moins que b              a < b;
Vrai, si a est plus que b               a > b;
Vrai, si a est moins ou égal à b        a <= b;
Vrai, si a est plus ou égal à b         a >= b;
```

Il ne faut pas comparer deux nombres réels à l'égalité l'un l'autre. Dans la plupart des cas, les deux nombres apparemment identiques peuvent être inégales à cause de la différence de la valeur dans le 15-ième signe après la virgule. Pour la comparaison correcte de deux nombres réels il faut comparer la différence normalisée de ces nombres à la valeur nulle.

Exemple:

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0) return(true);
   else return(false);
  }
void OnStart()
  {
   double first=0.3;
   double second=3.0;
   double third=second-2.7;
   if(first!=third)
     {
      if(CompareDoubles(first,third))
         printf("%.16f  %.16f   pourtant sont égaux",first,third);
     }
  }
// Résultat: 0.3000000000000000  0.2999999999999998   sont égaux
```

Voir aussi

Les priorités et l'ordre des opérations

© 2000-2015, MetaQuotes Software Corp.

Les opérations logiques

Négation Logique NON (!)

L' opérande de la négation logique NON (!) doit avoir le type arithmétique. Le résultat est égal au VRAI(1), si la valeur d'opérande est FAUX(0) et est égal au FAUX(0), si l'opérande n'est pas égal au FAUX(0).

```
if(!a) Print("pas 'a'");
```

Opération logique OU (||)

Une opération logique OU (||) des valeurs x et y. La valeur de l'expression est VRAI (1), si la valeur x ou y est vrai (pas le zéro). Au contraire c' est - FAUX (0).

```
if(x<0 || x>=max_bars) Print("out of range");
```

Opération logique ET (&&)

Une opération logique ET (&&) des valeurs x et y. La valeur de l'expression est VRAI (1), si les valeurs x et y sont vrais (pas le zéro). Au contraire c' est - FAUX (0). Les expressions logiques sont calculées complètement, c'est-à-dire le schéma ce qu'on appelle "une évaluation courte" ne s'applique pas à eux.

```
if(p!=x && p>y) Print("TRUE");
```

L'évalution courte des opérations logiques

On applique aux expressions logiques le schéma soi-disant de "l'évalution courte", c'est-à-dire, le calcul de l'expression cesse au moment où on peut exactement évaluer le résultat de l'expression.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- le premier exemple de l'évalution courte
   if(func_false() && func_true())
     {
      Print("L'opération && : vous ne verrez jamais ce message");
     }
   else
     {
      Print("L'opération && : Le résultat de la première expression est false, c'est p
     }
//---le deuxième exemple de l'évalution courte
   if(!func_false() || !func_true())
     {
      Print("L'opération II: Le résultat de la première expression est true, c'est pou
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   else
     {
      Print("L'opération ||: Vous ne verrez jamais ce message");
     }
  }
//+------------------------------------------------------------------+
//| la fonction rend toujours false                                  |
//+------------------------------------------------------------------+
bool func_false()
  {
   Print("La fonction func_false()");
   return(false);
  }
//+------------------------------------------------------------------+
//| la fonction rend toujours true                                   |
//+------------------------------------------------------------------+
bool func_true()
  {
   Print("La fonction func_true()");
   return(true);
  }
```

Voir aussi

Les priorités et l'ordre des opérations

© 2000-2015, MetaQuotes Software Corp.

Les opérations bit à bit

Complémentation à l'unité

La complémentation à l' unité de la valeur de variable. La valeur de l'expression contient 1 dans tous les chiffres, dans lesquels la valeur de variable contient 0, et 0 est dans tous les chiffres, dans lesquels la valeur de variable contient 1.

```
b = ~n;
```

Exemple:

```
   char a='a',b;
   b=~a;
   Print("a = ",a, "  b = ",b);
// Résultat  sera:
// a = 97   b = -98
```

Décalage à droite

La représentation binaire de x est déplacé à droite par les chiffres y. Si la valeur décalé a le type non signé, le décalage logique à droite s'effectue, c'est-à-dire, les catégories libérés à gauche seront remplis par les zéros.

Si la valeur décalé a le type de signe, le décalage arithmétique à droite s'effectue, c'est-à-dire, les catégories libérés à gauche seront remplis de la valeur du bit de signe (si le nombre est positif, alors la valeur du bit de signe est 0, si le nombre est négatif, la valeur du bit de signe est 1)

```
x = x >> y;
```

Exemple:

```
   char a='a',b='b';
   Print("Before:  a = ",a, "  b = ",b);
//--- faisons le décalage à droite
   b = a >> 1;
   Print("After:   a = ",a, "  b = ",b);
// Résultat sera:
// Before:  a = 97   b = 98
// After:   a = 97   b = 48
```

Décalage à gauche

La représentation binaire de x est décalé à gauche par les chiffres y; les catégories libérés à droite seront remplis par les zéros.

```
x = x << y;
```

Exemple:

```
   char a='a',b='b';
   Print("Before:  a = ",a, "  b = ",b);
//--- faisons le décalage à gauche
```

© 2000-2015, MetaQuotes Software Corp.

```
   b=a << 1;
   Print("After:   a = ",a, "  b = ",b);
// Résultat sera:
// Before:  a = 97   b = 98
// After:   a = 97   b = -62
```

Il ne faut pas faire le décalage au nombre de bits égal ou majeur, que la capacité de variable décalée, parce que le résultat de cette opération n'est pas défini.

Opération bit à bit ET

L'opération bit à bit ET des représentations binaires x et y. La valeur de l'expression contient 1 (VRAI) dans tous les chiffres, dans lesquels et x, et y ne contiennent pas le zéro et 0 (FAUX)dans tous les autres chiffres.

```
b = ((x & y) != 0);
```

Exemple:

```
   char a='a',b='b';
//--- opération ET
   char c=a&b;
   Print("a = ",a,"  b = ",b);
   Print("a & b = ",c);
// Résultat sera:
// a = 97   b = 98
// a & b = 96
```

Opération bit à bit OU

L'opération bit à bit OU des représentations binaires x et y. La valeur de l'expression contient 1 dans tous les chiffres, dans lesquels x ou y ne contiennent pas 0, et 0 - dans tous les autres chiffres.

```
b = x | y;
```

Exemple:

```
   char a='a',b='b';
//--- opération OU
   char c=a|b;
   Print("a = ",a,"  b = ",b);
   Print("a | b = ",c);
// Résultat sera:
// a = 97   b = 98
// a | b = 99
```

Opération bit à bit excluant OU

L'opération bit à bit excluant OU (eXclusive OR) des représentations binaires x et y. La valeur de l'expression contient 1 dans les chiffres, dans lesquels x et y ont les valeurs binaires différentes, et 0 - dans tous les autres chiffres.

```
b = x ^ y;
```

© 2000-2015, MetaQuotes Software Corp.

Exemple:

```
   char a='a', b='b';
//--- opération excluant OU
   char c=a^b;
   Print("a = ",a,"  b = ",b);
   Print("a ^ b = ",c);
// Résultat sera:
// a = 97   b = 98
// a ^ b = 3
```

Les opérations bit à bit se produisent seulement avec les nombres entiers.

Voir aussi

Les priorités et l'ordre des opérations

© 2000-2015, MetaQuotes Software Corp.

Les autres opérations

Indexation ([])

Quand on s'adresse au i-ème élément du tableau la valeur de l' expression est la valeur d'une variable avec le nombre ordinal i.

Exemple:

```
array[i] = 3; // Assigner la valeur au 3 i-ème élément du tableau array.
```

Seulement un nombre entier peut être l'index du tableau. Il est admis non plus que les tableaux quadrididimensionaux. L'indexation de chaque mesure est indexé de 0 **à la grandeur de mesure** -1. Dans le cas particulier d'un tableau unidimensionnel des 50 éléments la référence au premier élément ressemblera à array[0], au dernier élément - array[49].

En adressant au-delà du tableau le sous-système génère une erreur critique et l'exécution du programme sera arrêté.

Appel d'une fonction avec les arguments x1, x2,..., xn

Chaque argument peut être une constante, une variable ou une expression du type. Les arguments transmissibles sont séparés par les virgules et doivent être à l'intérieur des parenthèses, la parenthèse ouvrante doit suivre le nom de la fonction appelée.

La valeur d'expression est la valeur, rendue par la fonction. Si le type de retour de la fonction est void, alors l'appel d'une telle fonction ne peut pas être placé à droite dans l'opération d'attribution. Prêtez attention que que l'ordre de l'exécution des expression x1,..., xn est garanti.

Exemple:

```
   int length=1000000;
   string a="a",b="b",c;
//---
   int start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);
```

Opération le virgule (,)

Les expressions séparées par des virgules sont évalués de gauche à droite. Tous les effets accessoires du calcul d'expression gauche peuvent apparaître avant que le calcul de l'expression droite sera faite. Le type et la valeur de résultat coïncident avec le type et la valeur de l'expression. La liste des

© 2000-2015, MetaQuotes Software Corp.

paramètres transmissibles peut être considérée comme un exemple (voir ci-dessus).

Exemple:

```
for(i=0,j=99; i<100; i++,j--) Print(array[i][j]);
```

Opération le point (.)

Pour un accès direct aux membres publiques des structures et des classes on utilise l'opération le point. La syntaxe.

```
Nom_de la variable_de type_de structure.Nom_du membre
```

Exemple:

```
   struct SessionTime
     {
      string sessionName;
      int    startHour;
      int    startMinutes;
      int    endHour;
      int    endMinutes;
     } st;
   st.sessionName="Asian";
   st.startHour=0;
   st.startMinutes=0;
   st.endHour=9;
   st.endMinutes=0;
```

Opération de la résolution de contexte (::)

Chaque fonction dans le programme mql5 a son contexte d'exécution. Par exemple, une fonction du système Print() est effectuée dans un contexte mondial. Les fonctions importées sont appelées dans le cadre de l'importation. Les fonctions-les méthodes des classes ont le contexte de la classe correspondante. La syntaxe de l'opération de la résolution de contexte:

```
[Nom_de contexte]::Nom_de la fonction (les paramètres)
```

Si le nom du contexte est absent, c'est une référence évidente à l'utilisation du contexte mondial. S'il n'y a pas aucune opération de la résolution de contexte, la fonction est demandée dans le contexte immédiat. S'il n'y a pas aucune fonction dans le contexte local, la recherche est faite dans un contexte mondial.

En outre l'opération de la résolution de contexte est utiliséé pour la définition de fonction-membre de la classe.

```
type Nom_de_la_classe::Nom_de_la_fonction(description_des_paramètres)
   {
// corps de fonction
   }
```

© 2000-2015, MetaQuotes Software Corp.

Use of several functions of the same name from different execution contexts in a program may cause ambiguity. The priority order of function calls without explicit scope specification is the following:

1. Class methods. If no function with the specified name is set in the class, move to the next level.
2. MQL5 functions. If the language does not have such a function, move to the next level.
3. User defined global functions. If no function with the specified name is found, move to the next level.
4. Imported functions. If no function with the specified name is found, the compiler returns an error.

To avoid the ambiguity of function calls, always explicitly specify the function scope using the scope resolution operation.

Exemple:

```
#property script_show_inputs
#import "kernel32.dll"
   int GetLastError(void);
#import

class CCheckContext
  {
   int          m_id;
public:
                CCheckContext() { m_id=1234; }
protected:
   int          GetLastError() { return(m_id); }
  };
class CCheckContext2 : public CCheckContext
  {
   int          m_id2;
public:
                CCheckContext2() { m_id2=5678; }
   void         Print();
protected:
   int          GetLastError() { return(m_id2); }
  };
void CCheckContext2::Print()
  {
   ::Print("Terminal GetLastError",::GetLastError());
   ::Print("kernel32 GetLastError",kernel32::GetLastError());
   ::Print("parent GetLastError",CCheckContext::GetLastError());
   ::Print("our GetLastError",GetLastError());
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CCheckContext2 test;
   test.Print();
  }
//+------------------------------------------------------------------+
```

Opération de la prise de la grandeur du type de données ou la

© 2000-2015, MetaQuotes Software Corp.

grandeur de l'objet de n'importe quel type de données (sizeof)

A l'aide de l'opération sizeof on peut définir la grandeur de la mémoire qui correspond à l'identificateur ou le type. L'opération sizeof a le format suivant:

Exemple:

```
sizeof(expréssion)
```

A titre de l'expression on peut utiliser n'importe quel identificateur, ou le nom du type mis entre parenthèses. Il faut noter qu'on ne peut pas utiliser le nom de type void, et l'identificateur ne peut pas se rapporter au champ des bits ou être le nom de la fonction.

Si à titre de l'expression on indique le nom du tableau statique (c'est-à-dire, on donne la première dimension), le résultat est la grandeur de tout le tableau (c'est-à-dire le produit du nombre d'éléments sur la longueur du type). Si à titre de l'expression on indique le nom du tableau dynamique (la première dimension n'est pas donnée), le résultat sera la grandeur de l'objet du tableau dynamique.

Quand sizeof sont appliqués au nom du type structural ou de la classe ou vers l'identificateur ayant le type structural ou la classe, le résultat est la grandeur réelle de la structure ou de la classe.

Exemple:

```
struct myStruct
  {
   char   h;
   int    b;
   double f;
  } str;
Print("sizeof(str) = ",sizeof(str));
Print("sizeof(myStruct) = ",sizeof(myStruct));
```

Le calcul de la grandeur se passe à l'étape de la compilation.

Voir aussi

Les priorités et l'ordre des opérations

© 2000-2015, MetaQuotes Software Corp.

Les priorités et l'ordre des opérations

La priorité est la même pour chaque groupe d'opérations dans le tableau. Plus que la priorité du groupe des opérations est supérieur, plus haut il se place dans le tableau. L'ordre d'exécution détermine le groupement d'opérateurs et d'opérandes.

Attention: La priorité d'exécution des opérations dans le langage MQL5 correspond à la priorité, adoptée dans le langage C++, et diffère de la priorité donnée dans le langage MQL4.

Opération	Déscription	L'ordre d'exécution
() [] .	Appel de fonction Extraction d'élément du tableau Extraction d'élément de la structure	De gauche à droite
! ~ - ++ -- (type) sizeof	Négation logique Négation bit à bit (complement) Changement de signe Incrémentation par une unité (increment) Décrescence par une unité (decrement) Transformation du type Détermination de la grandeur en bytes	De droite à gauche
* / %	Multiplication Division Division de module	De gauche à droite
+ -	Addition Soustraction	De gauche à droite
<< >>	Décalage à gauche Décalage à droite	De gauche à droite
< <= > >=	Moins que Moins que ou égal Plus que Plus que ou égal	De gauche à droite
== !=	Egal Pas égal	De gauche à droite
&	Opération bit à bit ET	De gauche à droite
^	Opération bit à bit excluant OU (eXclude OR)	De gauche à droite
|	Opération bit à bit OU	De gauche à droite

© 2000-2015, MetaQuotes Software Corp.

&&	Opération logique ET	De gauche à droite
||	Opération logique OU	De gauche à droite
?:	Opération conditionnelle	De droite à gauche
= *= /= %= += -= <<= >>= &= ^= |=	Attribution Multiplication avec l' attribution Division avec l' attribution Division de module avec l' attribution Addition avec l' attribution La soustraction avec l' attribution Décalage à gauche avec l' attribution Décalage à droite avec l' attribution Bit à bit ET avec l' attribution Excluant OU avec l' attribution Bit à bit OU avec l' attribution	De droite à gauche
,	Virgule	De gauche à droite

Pour changer l'ordre d'exécution des opérations on utilise les parenthèse qui ont la plus haute priorité.

© 2000-2015, MetaQuotes Software Corp.

Les opérateurs

Les opérateurs du langage décrivent quelques opérations algorithmiques qui doivent être exécutées pour résoudre le problème. Le corps de programme est un ordre de tels opérateurs. Des opérateurs consécutifs sont séparés par un point -virgule.

Opérateur	Déscription
Opérateur composé {}	Un ou plusieurs opérateurs de n'importe quel type, qui sont dans les accolades { }
Opérateur d'expression (;)	N'importe quelle expression qui se termine par un point-virgule (;)
Opérateur return	Termine l'exécution de la fonction actuelle et retourne le contrôle au programme appelant
Opérateur conditionnel if-else	Utilisé lorsqu'il est nécessaire de faire un choix
Opérateur conditionnel ?:	Un analogue plus simple de l'opérateur conditionnel if-else
Opérateur de sélection switch	Transfère le contrôle à l'opérateur, qui correspond à la valeur d'expression
Opérateur de boucle while	Exécute l'opérateur jusqu'à l'expression vérifiable devienne fausse. La vérification de l'expression est faite avant chaque itération
Opérateur de boucle for	Exécute l'opérateur jusqu'à l'expression vérifiable devienne fausse. La vérification de l'expression est faite avant chaque itération
Opérateur de boucle do-while	Exécute l'opérateur jusqu'à l'expression vérifiable devienne fausse. L'épreuve de condition d'achèvement de boucle se produit à la fin, après chaque boucle. Le corps de la boucle est toujours exécuté au moins une fois.
Opérateur break	Termine l'exécution de l'opérateur externe attaché le plus proche switch, while, do-while et for
Opérateur continue	Passe le contrôle au début de l'opérateur plus proche externe de la boucle while, do-while ou for
Opérateur new	Crée un objet de la grandeur appropriée et rend le descripteur de l'objet créé.
Opérateur delete	Supprime l'objet créé par l'opérateur new

Un opérateur peut occuper une ou plusieurs lignes. Deux ou plusieurs opérateurs peuvent être situés sur une ligne. Les opérateurs qui contrôlent l'ordre de l'exécution (if, if-else, switch, while et for),

© 2000-2015, MetaQuotes Software Corp.

peuvent être imbriquées.

Exemple:

```
if(Month() == 12)
  if(Day() == 31) Print("Happy New Year!");
```

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'opérateur composé

Un opérateur composé (un bloc) se compose d'un ou plusieurs nombres des opérateurs de n'importe quel type qui sont dans les accolades { }. L'accolade finale ne doit pas être suivie par un point-virgule (;).

Exemple:

```
if(x==0)
  {
   Print("invalid position x = ",x);
   return;
  }
```

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'opérateur d'expression

Toute expression qui se termine par un point-virgule (;), est l'opérateur. Voici quelques exemples d'opérateurs d'expression.

Opérateur d'attribution:

L'identificateur est = une expression;

```
x=3;
y=x=3;
bool equal=(x==y);
```

L'opérateur d'attribution peut être utilisé plusieurs fois dans une expression. Dans ce cas, le traitement d'expression est fait de droite à gauche:

Opérateur d'appel de fonction:

Nom_de fonction (argument1,..., argumentN);

```
FileClose(file);
```

Opérateur Vide

Il se compose seulement d'un point-virgule (;) et est utilisé pour dénoter un corps vide d'un opérateur de contrôle.

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'opérateur de retour return

L'opérateur return termine l'exécution de fonction actuelle et rend le contrôle au programme appelant. Le résultat de calcul d'expression retourne par la fonction appelée. L'expression peut contenir un opérateur d'attribution.

Exemple:

```
int CalcSum(int x, int y)
  {
   return(x+y);
  }
```

Dans les fonctions du type de valeur de retour void il faut utiliser l'opérateur return sans expression:

```
void SomeFunction()
  {
   Print("Hello!");
   return;     // cet opérateur peut être supprimé
  }
```

L'accolade terminal de la fonction signifie l'exécution implicite de l'opérateur return sans expression.

On peut rendre les types simples, les structures simples, les indicateurs des objets. A l'aide de l'opérateur *return* on ne peut pas rendre les tableaux, les objets des classes, les variables comme les structures complexes.

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'opérateur conditionnel if-else

L'opérateur IF - ELSE s'utilise quand il faut faire le choix. Formellement, la syntaxe se présente comme:

```
if (expression)
     opérateur1
else
     opérateur2
```

Si l'expression est vraie, operator1 est exécuté et le contrôle est donné à l'opérateur qui suit operator2 (c'est-à-dire l'operateur2 n'est pas exécuté). Si l'expression est fausse, l'operateur2 est exécuté.

La partie else de l'operateur if peut être omise. Voilà pourquoi l'ambiguité peut être dans les opérateurs imbriqués if avec la partie omise else. Dans ce cas-là else connecte avec operateur précédent le plus proche if dans le même bloc, qui n'a pas la partie else.

Exemples:

```
//--- La partie else appartient au deuxième opérateur if:
if(x>1)
   if(y==2) z=5;
else     z=6;
//--- La partie else appartient au premier opérateur if
if(x>1)
  {
   if(y==2) z=5;
  }
else        z=6;
//--- Opérateurs imbriqués
if(x=='a')
  {
   y=1;
  }
else if(x=='b')
  {
   y=2;
   z=3;
  }
else if(x=='c')
  {
   y=4;
  }
else Print("ERROR");
```

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'opérateur conditionnel ?:

La forme générale de l'opérateur ternaire se présente comme:

```
expression1? expression2:expression3
```

En tant que le premier opérande - "expression1" - n'importe quelle expression peut être utilisé, dont le résultat est la valeur du type bool. Si le résultat est true, donc l'opérateur, commandé par le deuxième opérande, est exécutée, c'est-à-dire "expression2".

Si le premier opérande est false, dont le troisième opérande est exécutée - "expression3". Les deuxième et troisième opérandes, c'est-à-dire "expression2 " et "expression3", devraient rendre des valeurs d'un type et ne devrait pas avoir le type void. Le résultat de l'exécution d'opérateur conditionnel est le résultat d'expression2 ou le résultat de l'expression3, selon le résultat d'expression1.

```
//--- normalisons la différence entre les cours de clôture et ouverts pour une gamme c
double true_range = (High==Low)?0:(Close-Open)/(High-Low);
```

Cette entrée est équivalente à la suivante

```
   double true_range;
   if(High==Low)true_range=0;                // si High et Low sont égaux
   else true_range=(Close-Open)/(High-Low); // si la gamme n'est pas nulle
```

Les limitations de l'utilisation de l'opérateur

L'opérateur sur la base de la valeur "expression 1" doit rendre une des deux valeurs - ou "l'expression 2", ou "l'expression 3". Il y a une série de limitations à ces expressions:

1. Il ne faut pas confondre le type d'utilisateur avec le type simple ou l'énumération. Pourl'indicateur il est admissible d'utiliserNULL.
2. Si les types des valeurs sont simples, le type de l'opérateur sera le type maximal (regarde La conformation des types).
3. Si une des valeurs a le type l'énumération et le deuxième c'est le type numérique, l'énumération est remplacé sur int et la deuxième règle agit.
4. Si les deux valeurs sont les valeurs des énumérations, leurs types doivent être identiques, et le type de l'opérateur sera l'énumération.

Les limitations pour les types d'utilisateur (les classes ou les structures):

a) les types doivent être identiques ou un type doit être héritéde l'autre.

b) si les types ne sont pas identiques (l'héritage), le descendant est amené non évidemment au parent, c'est-à-dire le type de l'opérateur sera le type du parent.

c) il ne faut pas confondre l'objet et l'indicateur - ou les deux expressions sont les objets, ou les indicateurs. Pour l'indicateur il est admissible d'utiliserNULL.

Note

Soyez attentifs à l'utilisation de l'opérateur conventionnel à titre de l'argument de la fonction

© 2000-2015, MetaQuotes Software Corp.

surchargée,, puisque le type du résultat de l'opérateur conventionnel est défini au moment de la compilation du programme. Et ce type est défini comme le type le plus grand des types "expression2" et "expression3".

Exemple:

```
void func(double d) { Print("double argument: ",d); }
void func(string s) { Print("string argument: ",s); }

bool   Expression1=true;
double Expression2=M_PI;
string Expression3="3.1415926";

void OnStart()
  {
   func(Expression2);
   func(Expression3);

   func(Expression1?Expression2:Expression3);   // recevons la prévention du compilate
   func(!Expression1?Expression2:Expression3);  // recevons la prévention du compilate
  }

//   Résultat:
//   double argument: 3.141592653589793
//   string argument: 3.1415926
//   string argument: 3.141592653589793
//   string argument: 3.1415926
```

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'opérateur-aiguilleur switch

Il compare la valeur de l'expression avec des constantes dans toutes les variantes *case* et passe le contrôle à l'opérateur qui correspond à la valeur d'expression. Chaque variante de *case* peut être marquée par une constante entière, une constante symbolique ou une expression constante. Une expression constante ne peut pas inclure des variables ou des appels de fonction. L'expression de l'opérateur *switch* doit être du type entier.

```
switch(expression)
  {
   case constante: opérateurs
   case constante: opérateurs
      ...
   default: opérateurs
  }
```

Les opérateurs raccordés avec l'étiquette *default*, s'exécutent, si aucun des constantes dans les opérateurs *case* n'est pas égal à la valeur de l'expression. La variante *default* ne devrait pas nécessairement être déclarée et ne devrait pas nécessairement être la dernière. Si aucune des constantes ne correspond pas à la valeur de l'expression et la variante *default* est absent, aucune action ne s' exécute pas.

Le mot-clé *case* avec la constante sont tout simplement les étiquettes, et si les opérateurs seront exécutés pour une variante *case*, ensuite les opérateurs de toutes les versions subséquentes seront exécutés jusqu'à ce que l'opérateur *break* ne rencontrera pas, ce que permet d'attacher une séquence d'opérateurs avec plusieurs variantes.

Une expression constante est calculée pendant la compilation. Aucune de deux constantes dans un opérateur -aiguilleur ne peuvent avoir la même valeur.

Exemples:

```
//--- premier exemple
switch(x)
  {
   case 'A':
      Print("CASE A");
      break;
   case 'B':
   case 'C':
      Print("CASE B or C");
      break;
   default:
      Print("NOT A, B or C");
      break;
  }

//---  deuxième exemple
   string res="";
   int i=0;
   switch(i)
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       case 1:
          res=i;break;
       default:
          res="default";break;
       case 2:
          res=i;break;
       case 3:
          res=i;break;
      }
   Print(res);
/*
   Résultat
   default
*/
```

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'opérateur de boucle while

L' opérateur while se compose d'une expression vérifiable et de l'opérateur, qui doit être exécuté:

```
while(expression)
  opérateur;
```

Si l'expression est vraie, l'opérateur est exécuté jusqu'à ce que l'expression ne devienne fausse. Si l'expression est fausse, le contrôle se passe à l'opérateur suivant. La valeur d'expression est définie avant que l'opérateur est exécuté. Donc, si l'expression est fausse dès le début, l'opérateur ne sera pas exécuté du tout.

Note

S'il est supposé de traiter une grande quantité d'itérations dans la boucle, il est recommandé de contrôler le fait de l'achèvement forcé du programme à l'aide de la fonctionIsStopped().

Exemple:

```
while(k<n  && !IsStopped())
  {
   y=y*x;
   k++;
  }
```

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'opérateur de boucle for

L'opérateur for se compose de trois expressions et d'un opérateur exécuté:

```
for(expression1; expression2; expression3)
   opérateur;
```

L'expression1 décrit la boucle de l'initialisation. L'expression2 vérifie les conditions de l'accomplissement de la boucle. S'il est vrai, l'opérateur du corps de boucle for exécuté. Tout se répète, avant que l'expression2 ne devient pas fausse. Si elle est fausse, la boucle se termine et le contrôle se transfère à l'opérateur suivant. L'expression2 évaluée après chaque itération.

L'opérateur for est équivalente à la séquence suivante d'opérateurs:

```
expression1;
 while(expression2)
   {
    opérateur;
    expression3;
   };
```

Chacune des trois expressions ou toutes les trois expressions dans l'opérateur for peuvent être absentes, toutefois, les points-virgules (;) qui les sèparent ne peuvent être omis. Si expression2 est omise, il est considéré comme toujours vraie. L'opérateur for(;;) est une boucle infinie équivalente à l'opérateur while(1). Chacune des expressions1 et des expressions3 peut être composée des plusieurs expressions, combinées par un opérateur de virgule ','.

Note

S'il est supposé de traiter une grande quantité d'itérations dans la boucle, il est recommandé de contrôler le fait de l'achèvement forcé du programme à l'aide de la fonctionIsStopped().

Exemples:

```
for(x=1;x<=7000; x++)
   {
    if(IsStopped())
       break;
    Print(MathPower(x,2));
   }
//--- autre exemple
for(;!IsStopped();)
   {
    Print(MathPower(x,2));
    x++;
    if(x>10) break;
   }
//--- troisième exemple
for(i=0,j=n-1;i<n && !IsStopped();i++,j--) a[i]=a[j];
```

Voir aussi

© 2000-2015, MetaQuotes Software Corp.

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'opérateur de boucle do while

Les boucles for et while font le contrôle au début d'une boucle et pas à la fin d'une boucle. Le troisième opérateur de boucle do - while vérifie la condition du bout à la fin, après chaque itération de boucle. Le corps de boucle est toujours exécuté au moins une fois.

```
do
   opérateur;
while(expression);
```

D'abord l'opérateur est exécuté, après l'expression est calculée. Si elle est vraie, alors l'opérateur est exécutée encore et etc. Si l'expression est fausse, la boucle se termine.

Note

S'il est supposé de traiter une grande quantité d'itérations dans la boucle, il est recommandé de contrôler le fait de l'achèvement forcé du programme à l'aide de la fonctionIsStopped().

Exemple:

```
//--- calcul de la séquence des nombres de Fibonacci
   int counterFibonacci=15;
   int i=0,first=0,second=1;
   int currentFibonacciNumber;
   do
     {
      currentFibonacciNumber=first+second;
      Print("i = ",i,"  currentFibonacciNumber = ",currentFibonacciNumber);
      first=second;
      second=currentFibonacciNumber;
      i++; // boucle sera infinie sans cet opérateur!
     }
   while(i<counterFibonacci && !IsStopped());
```

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'opérateur d'accomplissement break

L'opérateur break termine l'exécution de l'opérateur extérieur niché le plus proche switch, while, do-while ou for. Le contrôle est passé à l'opérateur qui suit l'opérateur qui de termine. L'un des objectifs de cet opérateur c'est finir l'exécution de la boucle lorsqu'on assigne une valeur d'une variable particulière.

Exemple:

```
//--- recherche le premier élément zéro
for(i=0;i<array_size;i++)
  if(array[i]==0)
    break;
```

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

Opérateur d' extension continue

L'opérateur continue transmet la gestion au début de l'opérateur plus proche extérieur de la boucle while, do-while ou for, en provoquant le début de l'itération suivante. Cet opérateur est opposé par l'action à l'opérateur break.

Exemple:

```
//--- somme de tous les éléments non zéro
int func(int array[])
  {
   int array_size=ArraySize(array);
   int sum=0;
   for(int i=0;i<array_size; i++)
     {
      if(a[i]==0) continue;
      sum+=a[i];
     }
   return(sum);
  }
```

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L' opérateur de la génération de l'objet new

L'opérateur new crée automatiquement un objet de la valeur appropriée, appelle le constructeur de l'objet et retourne un descripteur de l'objet créé le descripteur de l' objet créé. En cas de l'échec, l'opérateur rend un descripteur nul, qui peut être comparé avec la constante NULL.

L'opérateur *new* peut être appliqué seulement aux objets de la classe, il ne peut pas être appliqué aux structures.

L'opérateur ne s'applique pas pour la génération des tableaux des objets. Pour ça il faut utiliser la fonction ArrayResize().

Exemple:

```
//+------------------------------------------------------------------+
//| génération de la figure                                          |
//+------------------------------------------------------------------+
void CTetrisField::NewShape()
  {
   m_ypos=HORZ_BORDER;
//--- on crée une des 7 formes possibles par hasard
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- on dessine
   if(m_shape!=NULL)
     {
      //--- établissements initiaux
      m_shape.SetRightBorder(WIDTH_IN_PIXELS+VERT_BORDER);
      m_shape.SetYPos(m_ypos);
      m_shape.SetXPos(VERT_BORDER+SHAPE_SIZE*8);
      //--- on dessine
      m_shape.Draw();
     }
//---
  }
```

Il faut noter que le descripteur de l'objet n'est pas un pointeur à la mémoire.

L'objet créé par l'opérateur new, doit être explicitement détruits par l'opérateur delete.

Voir aussi

© 2000-2015, MetaQuotes Software Corp.

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'opérateur de la suppression de l'objet delete

L'opérateur delete supprime l'objet, crée par l'opérateur new, appelle le destructeur de la classe correspondante et libére la mémoire occuppée par l'objet. On utilise le descripteur réel de l'objet existant comme opérande. Apres l'exécution de l'opération delete le descripteur de l'objet devient invalide.

Exemple:

```
      //--- supprimer une figure emballée
      delete m_shape;
      m_shape=NULL;
      //--- créer une nouvelle figure
      NewShape();
```

Voir aussi

Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

Les fonctions

Chaque tâche peut être divisée en sous-tâches, dont chacun peut être soit directement représenté sous forme de code, ou même divisée en petites sous-tâches. Cette méthode est appelée *un affinement pas à pas*. Les fonctions sont utilisées pour écrire le code programmatique de ces sous-tâches résolues. Le code qui décrit ce que fait une fonction appelle *la définition de fonction:*

```
titre_de fonction
  {
   instructions
  }
```

Tout ce qui est avant la première accolade est *le titre* d'une définition de la fonction, et ce qui est entre les accolades est *le corps* d'une définition de la fonction. Le titre de fonction comprend une description du type de retour, le nom de (l'identificateur) et des paramètres formels. Le nombre de paramètres passés à la fonction est limitée et ne peut excéder 64.

La fonction peut être appelée des autres parties du programme tant de fois que nécessaire. En fait, le type de retour, l'identificateur de fonction et les types des paramètres constituent *le prototype de fonction*.

Le prototype de la fonction - c'est la déclaration de fonction, mais pas sa définition. En raison de la déclaration explicite du type de retour et d'une liste de types d'argument, dans les appels aux fonctions une vérification stricte des types et les transformations implicites des types sont possibles. Très souvent les déclarations de fonction sont utilisées dans les classes pour améliorer la lisibilité du code.

La définition de fonction doit exactement correspondre à sa déclaration. Chaque fonction déclarée doit être définie.

Exemple:

```
double                          // type de valeur de retour
linfunc (double a, double b) // nom de fonction et la liste des paramètres
  {
                                // opérateur composé
   return (a + b);              // valeur de retour
  }
```

L'opérateur return peut rendre la valeur de retour de l'expression qui est dans cet opérateur. Si nécessaire la valeur de l'expression se convertit en type du résultat de fonction. On peut rendre les types simples, les structures simples, les indicateurs des objets. A l'aide de l'opérateur *return* on ne peut pas rendre les tableaux, les objets des classes, les variables comme les structures complexes.

Une fonction qui ne renvoie aucune valeur, devrait être décrite comme ayant le type void.

Exemple:

```
void errmesg(string s)
  {
   Print("error: "+s);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Les paramètres passés à la fonction peuvent avoir des valeurs implicites, qui sont définies par les constantes du type correspondant.

Exemple:

```
int somefunc(double a,
             double d=0.0001,
             int n=5,
             bool b=true,
             string s="passed string")
  {
   Print("paramètre a été transmis: d=",d," n=",n," b=",b," s=",s);
   return(0);
  }
```

Si on a assigné une valeur implicite à aucun paramètre,tous les paramètres suivants doivent aussi avoir des valeurs implicites.

Exemple de déclaration incorrecte:

```
int somefunc(double a,
             double d=0.0001,    // valeur déclarée par défaut 0.0001
             int n,              // valeur n'est pas déclarée par défaut !
             bool b,             // valeur n'est pas déclarée par défaut !
             string s="passed string")
  {
  }
```

Voir aussi

Surcharge, Fonctions virtuelles, Polymorphisme

© 2000-2015, MetaQuotes Software Corp.

L'appel de fonction

Si un nom qui n'a pas été décrit auparavant, apparaît dans l'expression et la suivie la parenthèse gauche, il est considéré comme le nom d'une fonction selon le contexte.

```
nom_de la fonction(x1, x2,..., xn)
```

Les arguments (les paramètres formels) sont passés par la valeur, c'est-à-dire chaque expression x1, . . . , xn est évaluée et la valeur se transmise à la fonction. L'ordre d'évaluation d'expressions et l'ordre du lancement des valeurs ne sont pas garantis. Pendant l'exécution on fait la vérification de nombre et des types des arguments qui étaient transmis par la fonction. Une telle façon de s'adresser à la fonction s'appelle un appel de valeur.

Un appel de fonction est une expression dont la valeur est la valeur de retour. Un type de fonction décrit doit correspondre au type de la valeur de retour. La fonction peut être déclarée ou décrite dans n'importe quelle partie du programme au niveau globale, c'est-à-dire, en dehors d'autres fonctions. La fonction ne peut pas être déclarée ou décrite à l'intérieur d'une autre fonction.

Exemples:

```
int start()
  {
   double some_array[4]={0.3, 1.4, 2.5, 3.6};
   double a=linfunc(some_array, 10.5, 8);
   //...
  }
double linfunc(double x[], double a, double b)
  {
   return (a*x[0] + b);
  }
```

Lorsqu'on appelle la fonction avec les paramètres implicites, la liste des paramètres transmis peut être limitée, mais pas avant le premier paramètre implicite.

Exemples:

```
void somefunc(double init,
              double sec=0.0001, //on a défini les valeurs par défaut
              int level=10);
//...
somefunc();                      // appel incorrecte . premier paramètre obligatoire
somefunc(3.14);                  // appel correcte
somefunc(3.14,0.0002);           // appel correcte
somefunc(3.14,0.0002,10);        // appel correcte
```

En appelant une fonction, on ne peut ne pas passer des paramètres, même avec des valeurs par défaut :

```
somefunc(3.14, , 10);           // appel incorrecte -> deuxième paramètre est omis.
```

Voir aussi

© 2000-2015, MetaQuotes Software Corp.

Surcharge, Fonctions virtuelles, Polymorphisme

© 2000-2015, MetaQuotes Software Corp.

La transmission des paramètres

Il y a deux méthodes, par lesquelles le langage de machine peut passer des arguments à un sous-programme (les fonctions). La première méthode est la transmission par la valeur. Cette méthode copie la valeur de l'argument dans un paramètre formel de fonction. Voilà pourquoi toute modification de ce paramètre dans la fonction n'a aucune influence sur l'argument correspondant de l'appel.

```
//+------------------------------------------------------------------+
//| transmission des paramètres selon la valeur                      |
//+------------------------------------------------------------------+
double FirstMethod(int i,int j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a et b avant l'appel: ",a," ",b);
   double d=FirstMethod(a,b);
   Print("a et b après l'appel: ",a," ",b);
  }
//--- résultat de l'exécution du script
//  a et b avant l'appel: 14 8
//  a et b après l'appel: 14 8
```

La seconde méthode est la transmission de l'argument par la référence. Dans ce cas, la référence au paramètre (et non pas sa valeur) est passée au paramètre de fonction. A l'intérieur de la fonction, il est utilisé pour s'adresser au paramètre réel spécifié dans l'appel. Cela signifie que les changements de paramètre influeront à l'argument utilisé pour appeler la fonction.

```
//+------------------------------------------------------------------+
//| transmission des paramètres selon la référence                   |
//+------------------------------------------------------------------+
double SecondMethod(int &i,int &j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
```

© 2000-2015, MetaQuotes Software Corp.

```
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a et b avant l'appel: ",a," ",b);
   double d=SecondMethod(a,b);
   Print("a et b après l'appel: ",a," ",b);
  }
//+------------------------------------------------------------------+
//--- résultat de l'exécution du script
//  a et b avant l'appel: 14 8
//  a et b après l'appel: 28 4
```

MQL5 utilise les deux méthodes, à une exception: les tableaux, les variables du type de structure et les objets de classe sont toujours transmis par la référence. Pour exclure les changements des paramètres réels (les arguments passés à l'appel de fonction) il faut utiliser le spécificateur d'accès const. Lorsqu' on essaye de modifier le contenu d'une variable déclarée avec le spécificateur *const*, le compilateur produira une erreur.

Note

Il faut noter que les paramètres sont passés à une fonction dans l'ordre inversé, c'est-à-dire, d'abord le dernier paramètre est calculé et transmis et ensuite l'avant-dernier etc. Le paramètre qui est le premier après les crochets, est calculé et transmis le dernier à son tour.

Exemple:

```
void OnStart()
  {
//---
   int a[]={0,1,2};
   int i=0;

   func(a[i],a[i++],"Premier appel (i = "+string(i)+")");
   func(a[i++],a[i],"Deuxième appel (i = "+string(i)+")");
// Résultat:
// Premier appel(i = 0) : par1 = 1     par2 = 0
// Deuxième appel(i = 1) : par1 = 1     par2 = 1

  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void func(int par1,int par2,string comment)
```

© 2000-2015, MetaQuotes Software Corp.

```
{
 Print(comment,": par1 = ",par1,"    par2 = ",par2);
}
```

Au premier appel dans l'exemple mentionné au début la variable ***i*** participe à la concaténation des chaînes.

```
"Premier appel(i="+string(i)+")"
```

En même temps sa valeur ne change pas. Ensuite, la variable **i** participe dans le calcul de l'élément de tableau ***a[i++]***, c'est-à-dire, après la prise un i élément de tableau la variable ***i*** s'incrimine. Et seulement après cela le premier paramètre avec la valeur changée de la variable ***i*** est calculée.

Au deuxième appel en calculant tous les trois paramètres, on utilise la même valeur i, modifiée lors du premier appel, et seulement après le calcul du premier paramètre la variable ***i*** se change de nouveau.

Voir aussi

Portée de visibilité et la durée de vie des variables, Surcharge, Fonctions virtuelles, Polymorphisme

© 2000-2015, MetaQuotes Software Corp.

La surcharge des fonctions

D'habitude le nom de fonction a tendance à refléter son but principal. Généralement, les programmes lisibles contiennent les identificateurs différents et bien choisis. Parfois les fonctions différentes sont utilisées pour les mêmes buts. Considérons, par exemple, une fonction qui calcule la valeur moyenne d'un tableau de nombres de précision doubles et la même fonction, mais qui opère d'un tableau des nombres entiers. On peut appeler tous les deux AverageFromArray:

```
//+------------------------------------------------------------------+
//| calcul de la moyenne pour un tableau du type double              |
//+------------------------------------------------------------------+
double AverageFromArray(const double & array[],int size)
  {
   if(size<=0) return 0.0;
   double sum=0.0;
   double aver;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];    // addition pour double
     }
   aver=sum/size;       // on divise juste la somme par le nombre
//---
   Print("Calcul de la moyenne pour un tableau du type double");
   return aver;
  }
//+------------------------------------------------------------------+
//| calcul de la moyenne pour un tableau du type int                 |
//+------------------------------------------------------------------+
double AverageFromArray(const int & array[],int size)
  {
   if(size<=0) return 0.0;
   double aver=0.0;
   int sum=0;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];     // addition pour int
     }
   aver=(double)sum/size;// donnons la somme au type double et divisons
//---
   Print("Calcul de la moyenne pour un tableau du type int");
   return aver;
  }
```

Chaque fonction contient le message de sortie via la fonction Print();

```
   Print("Calcul de la moyenne pour un tableau du type int");
```

Le compilateur choisit la fonction nécessaire en conformité avec les types des arguments et de leur

© 2000-2015, MetaQuotes Software Corp.

quantité. La règle, selon laquelle le choix est fait, s'appelle *un algorithme de correspondance à la signature*. Une signature est une liste de types utilisés dans la déclaration de fonction.

Exemple:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int    a[5]={1,2,3,4,5};
   double b[5]={1.1,2.2,3.3,4.4,5.5};
   double int_aver=AverageFromArray(a,5);
   double double_aver=AverageFromArray(b,5);
   Print("int_aver = ",int_aver,"   double_aver = ",double_aver);
  }
//--- Résultat du travail du script
// Calcul de la moyenne pour un tableau du type int
// Calcul de la moyenne pour un tableau du type double
// int_aver = 3.00000000    double_aver = 3.30000000
```

La surcharge est la pratique d'allouer des valeurs multiples à une fonction. La sélection d'une valeur précise dépend des types d'arguments reçus par la fonction. Une fonction spécifique est choisie en fonction de la correspondance de la liste d'arguments en appelant la fonction, à la liste de paramètres dans la déclaration de fonction.

Quand une fonction surchargée est appelée le compilateur doit avoir un algorithme permettant de choisir la fonction appropriée. L'algorithme qui exécute ce choix dépend de celui-là les transformations de quel type sont présentes. La meilleure corrélation doit être unique. Ellel doit être la meilleure au moins pour un argument ainsi que toutes les autres correspondances, pour tous les autres arguments.

Ci-dessous est un algorithme correspondant pour chaque argument.

Algorithme de choix de la fonction surchargée

1. Utiliser la correspondance stricte (si c' est possible).
2. Essayer l'accroissement standard du type.
3. Essayer la transformation standard du type.

L'augmentation standard du type est mieux que d'autres transformations standards. L'augmentation c'est la transformation float au double, ainsi à bool, char, short ou enum à int. En outre les transformations des tableaux similaires aux types entiers se rapportent aux transformations standards. Les types similaires sont: bool, char, uchar, tous les trois types sont des entiers d'un byte; les entiers de 2 bytes short et ushort; les entiers de 4 bytes int, uint et color; long, ulong et datetime.

Évidemment la correspondance stricte est la meilleure. Pour parvenir à cette correspondance les conformations peuvent être utilisées. Le compilateur ne peut pas faire face aux situations ambigues. Donc il ne faut pas compter sur les différences subtiles des types et des transformations implicites qui rendent la fonction surchargée ambigue.

© 2000-2015, MetaQuotes Software Corp.

Si vous doutez, utilisez la transformation explicite pour garantir la correspondence stricte.

Exemples de fonctions surchargées dans MQL5 vous pouvez voir à l' exemple de fonctions ArrayInitialize().

Les règles de surcharges des fonctions s'applique à la surcharge des méthodes de classe.

La surcharge des fonctions du système est autorisée, mais il faut surveiller que le compilateur puisse sélectionner exactement la fonction nécessaire. Par exemple, on peut surcharger la fonction de système fmax() par 3 façons différentes, mais deux variants serront correctes.

Exemple:

```
// surcharge est admissible elle se diffère du nombre de paramètres
double fmax(double a,double b,double c);

// surcharge avec l'erreur
// nombre de paramètres est différent, mais le dernier a une valeur par défaut
// ce qui conduit à la dissimulation de la fonction du système lors de l'appel, ce qui
double fmax(double a,double b,double c=DBL_MIN);

// surcharge normale selon le type de paramètre
int fmax(int a,int b);
```

Voir aussi

Surcharge, Fonctions virtuelles, Polymorphisme

© 2000-2015, MetaQuotes Software Corp.

Surcharge des opérations

Pour le confort de la lecture et l'écriture du code on permet la surcharge de certaines opérations. L'opérateur de la surcharge s'inscrit à l'aide du mot-clé operator. La surcharge des opérations suivantes est permise:

- binaires +,-,/,*,%,<<,>>,==,!=,<,>,<=,>=,=,+=,-=,/=,*=,%=,&=,|=,^=,<<=,>>=,&&,||,&,|,^;
- unaires +,-,++,--,!,~;
- l'opérateur de l'affectation =;
- l'opérateur de l'indexation [].

La surcharge des opérations permet d'utiliser la notation opérationnelle (l'enregistrement en forme des expressions simples) vers les objets complexes - les structures et les classes. L'enregistrement des expressions avec l'utilisation des opérations surchargées simplifie la perception du code initial, puisque la réalisation la plus complexe est caché.

Comme l'exemple examinons les nombres complexes les plus utilisés en mathématique, qui comprennent les parties valable et imaginaire. Dans la langue MQL5 il n'y a pas de type des données pour la représentation des nombres complexes, mais il y a la possibilité de créer un nouveau type des données dans l'aspect de la structure ou de la classe. Déclarons la structure complex et y définissons quatre méthodes qui réalisent quatre opérations arithmétiques:

```
//+------------------------------------------------------------------+
//| La structure pour les opérations avec les nombres complexes      |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // la partie valable
   double            im; // la partie imaginaire
   //--- les constructeurs
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- les opérations arithmétiques
   complex           Add(const complex &l,const complex &r) const;  // addition
   complex           Sub(const complex &l,const complex &r) const;  // soustraction
   complex           Mul(const complex &l,const complex &r) const;  // multiplication
   complex           Div(const complex &l,const complex &r) const;  // division
  };
```

Maintenant, nous pouvons déclarer des variables dans le code propre, qui représentent les nombres complexes et travailler avec eux.

Par exemple:

```
void OnStart()
  {
//--- déclarons et initialisons les variables d'un type complexe
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- sommons deux nombres
   complex z;
   z=a.Add(a,b);
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- multiplions deux nombres
   z=a.Mul(a,b);
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//---divisons deux nombres
   z=a.Div(a,b);
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
```

Mais il serait plus confortable pour les opérations habituelles arithmétiques avec nombres complexe d'utiliser les opérateurs habituels "+","-","*" et "/".

Le mot-clé operator est utilisé pour déterminer la fonction -membre qui effectue le changement du type. Les opérations unaires et binaires pour les variables-objets de la classe peuvent être surchargées comme les fonctions-membres non statiques. Ils agissent non évidemment sur l'objet de la classe.

La plupart des opérations binaires peuvent être surchargées comme les fonctions ordinaires qui prennent un ou deux arguments en forme de la variable de la classe ou en forme de l'indicateur sur l'objet de la classe donnée. Pour notre type complex la surcharge dans l'annonce aura un tel aspect:

```
   //--- les opérateurs
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
```

L'exemple complet du script:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- annonçons et initialisons les variables du type complexe
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
   //a.re=5;
   //a.im=1;
   //b.re=-1;
   //b.im=-5;
//---sommons deux nombres
   complex z=a+b;
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- multiplions deux nombres

   z=a*b;
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//---divisons deux nombres
   z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
//+------------------------------------------------------------------+
//| La structure pour les opérations avec les nombres complexes      |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // la partie réelle
   double            im; // la partie imaginaire
   //--- les constructeurs
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- les opérations arithmétiques
   complex           Add(const complex &l,const complex &r) const;  // l'addition
   complex           Sub(const complex &l,const complex &r) const;  // la soustractio
   complex           Mul(const complex &l,const complex &r) const;  // la multiplicat
   complex           Div(const complex &l,const complex &r) const;  // la division
   //--- les opérateurs binaires
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
  };
//+------------------------------------------------------------------+
//| L'addition                                                       |
//+------------------------------------------------------------------+
complex complex::Add(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re+r.re;
   res.im=l.im+r.im;
//---le résultat
   return res;
  }
//+------------------------------------------------------------------+
//| La soustraction                                                  |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
complex complex::Sub(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re-r.re;
   res.im=l.im-r.im;
//--- le résultat
   return res;
  }
//+------------------------------------------------------------------+
//| La multiplication                                                |
//+------------------------------------------------------------------+
complex complex::Mul(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re*r.re-l.im*r.im;
   res.im=l.re*r.im+l.im*r.re;
//--- le résultat
   return res;
  }
//+------------------------------------------------------------------+
//| La division                                                      |
//+------------------------------------------------------------------+
complex complex::Div(const complex &l,const complex &r) const
  {
//--- le nombre complexe vide
   complex res(EMPTY_VALUE,EMPTY_VALUE);
//--- la vérification au zéro
   if(r.re==0 && r.im==0)
     {
      Print(__FUNCTION__+": number is zero");
      return(res);
     }
//--- les variables auxiliaires
   double e;
   double f;
//--- le choix de la variante du calcul
   if(MathAbs(r.im)<MathAbs(r.re))
     {
      e = r.im/r.re;
      f = r.re+r.im*e;
      res.re=(l.re+l.im*e)/f;
      res.im=(l.im-l.re*e)/f;
     }
   else
     {
      e = r.re/r.im;
      f = r.im+r.re*e;
      res.re=(l.im+l.re*e)/f;
      res.im=(-l.re+l.im*e)/f;
     }
//--- le résultat
   return res;
  }
```

La plupart des opérations unaires peuvent être surchargées comme les fonctions ordinaires qui prennent un seul argument -l'objet de la classe ou l'indicateur sur lui. Ajoutons la surcharge des

© 2000-2015, MetaQuotes Software Corp.

opérations unaires "-" et "!".

```
//+------------------------------------------------------------------+
//| La structure pour les opérations avec les nombres complexes      |
//+------------------------------------------------------------------+
struct complex
  {
   double            re;       // la partie réelle
   double            im;       // la partie imaginaire
...
   //--- les opérateurs unaires
   complex operator-()  const; // le moins unaire
   bool    operator!()  const; // la négation
  };
...
//+------------------------------------------------------------------+
//| La surcharge de l'opérateur "Le moins unaire"                    |
//+------------------------------------------------------------------+
complex complex::operator-() const
  {
   complex res;
//---
   res.re=-re;
   res.im=-im;
//--- le résultat
   return res;
  }
//+------------------------------------------------------------------+
//| La surcharge de l'opérateur "La négation logique"                |
//+------------------------------------------------------------------+
bool complex::operator!() const
  {
//--- la partie réelle et imaginaire du nombre complexe sont égaux au zéro ?
   return (re!=0 && im!=0);
  }
```

Maintenant nous pouvons vérifier la valeur du nombre complexe au zéro et recevoir la valeur négative:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- annonçons et initialisons les variables du type complexe
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- diviserons deux nombres
   complex z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//--- le nombre complexe est égal par défaut au zéro (en constructeur par défaut re==0
   complex zero;
   Print("!zero=",!zero);
//--- attribuons une valeur négative
   zero=-z;
   PrintFormat("z=%.2f+i*%.2f,  zero=%.2f+i*%.2f",z.re,z.im, zero.re,zero.im);
   PrintFormat("-zero=%.2f+i*%.2f",-zero.re,-zero.im);
//--- vérifions encore une fois sur l'égalité au zéro
   Print("!zero=",!zero);
//---
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
```

Prêtez attention que nous n'avons pas la nécessité de surcharger l'opératation de l'affectation "=", puisqu'on peut copierles structures des types simplesles uns aux autres directement. Ainsi, maintenant nous pouvons écrire le code pour les calculs avec la participation des nombres complexes à la manière habituelle.

La surcharge de l'opérateur de l'indexation permet de recevoir les valeurs des tableaux conclus à l'objet, par le moyen plus simple et habituel, et cela contribue aussi à la meilleure lisibilité et la compréhension du code initial des programmes. Par exemple, il nous est nécessaire d'assurer l'accès au symbole dans la ligne selon la position indiquée. La chaîne dans la langue MQL5 est le type séparé string, qui n'est pas le tableau des symboles, mais à l'aide de l'opération surchargée de l'indexation dans la classe créée CString on peut assurer le travail simple et transparent:

```
//+------------------------------------------------------------------------------
//| La classe pour l'accès aux symboles dans la chaîne comme dans le tableau des symbo
//+------------------------------------------------------------------------------
class CString
  {
   string            m_string;

public:
                     CString(string str=NULL):m_string(str) { }
   ushort operator[] (int x) { return(StringGetCharacter(m_string,x)); }
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- le tableau pour la réception des symboles de la chaîne
   int     x[]={ 19,4,18,19,27,14,15,4,17,0,19,14,17,27,26,28,27,5,14,
                 17,27,2,11,0,18,18,27,29,30,19,17,8,13,6 };
   CString str("abcdefghijklmnopqrstuvwxyz[ ]CS");
   string  res;
//--- composons la phrase, ayant pris les symboles de la variable str
   for(int i=0,n=ArraySize(x);i<n;i++)
     {
      res+=ShortToString(str[x[i]]);
     }
//--- déduisons le résultat
   Print(res);
  }
```

Un autre exemple de la surcharge de l'opération de l'indexation - c'est le travail avec les matrices. La matrice représente un tableau bidimensionnel dynamique, la taille des tableaux ne sont pas définies d'avance. C'est pourquoi on ne peut pas annoncer le tableau de l'aspect array[][] sans indication de la deuxième dimension et puis transmettre ce tableau à titre du paramètre. La classe spéciale CMatrix, qui comprend le tableau des objets de la classe CRow peut servir de la sortie.

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- les opérations de l'addition et la multiplication des matrices
   CMatrix A(3),B(3),C();
//---préparons les tableaux pour les chaînes
   double a1[3]={1,2,3}, a2[3]={2,3,1}, a3[3]={3,1,2};
   double b1[3]={3,2,1}, b2[3]={1,3,2}, b3[3]={2,1,3};
//---remplissons les matrices
   A[0]=a1; A[1]=a2; A[2]=a3;
   B[0]=b1; B[1]=b2; B[2]=b3;
//--- déduisons les matrices dans le journal "Experts"
   Print("----les éléments de la matrice A");
   Print(A.String());
   Print("---- les éléments de la matrice B");
   Print(B.String());

//--- l'addition des matrices
   Print("---- l'addition des matrices A et B");
   C=A+B;
//--- la sortie de la représentation formatée de la chaîne
   Print(C.String());

//--- la multiplication des matrices
   Print("---- le produit des matrices A et B");
   C=A*B;
   Print(C.String());

//---et maintenant montrons comment recevoir les valeurs à la façon des tableaux dynar
   Print("Déduisons les valeurs de la matrice C par élément");
//--- trions dans le cycle de la ligne de la matrice - les objets CRow
   for(int i=0;i<3;i++)
     {
      string com="| ";
      //--- formons pour la valeur les chaînes de la matrice
      for(int j=0;j<3;j++)
        {
         //---recevons l'élément de la matrice selon les numéros de la chaîne et de la
         double element=C[i][j];// [i] - l'accès vers CRow dans le tableau m_rows[] ,
                                // [j] - l'opérateur surchargé de l'indexation dans CR
         com=com+StringFormat("a(%d,%d)=%G ; ",i,j,element);
        }
      com+="|";
      //--- déduisons les valeurs de la chaîne
      Print(com);
     }
  }
//+------------------------------------------------------------------+
//| La classe "La chaîne"                                            |
//+------------------------------------------------------------------+
class CRow
  {
private:
   double            m_array[];
public:
   //--- les constructeurs et le destructeur
                     CRow(void)          { ArrayResize(m_array,0);     }
                     CRow(const CRow &r) { this=r;                     }
                     CRow(const double &array[]);
```

© 2000-2015, MetaQuotes Software Corp.

```
                    ~CRow(void){};
   //---le nombre d'éléments dans la chaîne
   int               Size(void) const    { return(ArraySize(m_array));}
   //--- rend la chaîne avec des valeurs
   string            String(void) const;
   //--- l'opérateur de l'indexation
   double            operator[](int i) const  { return(m_array[i]);   }
   //---les opérateurs de l'affectation
   void              operator=(const double  &array[]); // le tableau
   void              operator=(const CRow & r);         // un autre objet CRow
   double            operator*(const CRow &o);          // l'objet CRow pour la multip
  };
//+------------------------------------------------------------------+
//| Le constructeur pour initialiser la chaîne par un tableau        |
//+------------------------------------------------------------------+
void  CRow::CRow(const double &array[])
  {
   int size=ArraySize(array);
//--- si le tableau n'est pas vide
   if(size>0)
     {
      ArrayResize(m_array,size);
      //--- remplissons par les valeurs
      for(int i=0;i<size;i++)
         m_array[i]=array[i];
     }
//---
  }
//+------------------------------------------------------------------+
//| L'opération de l'affectation pour le tableau                     |
//+------------------------------------------------------------------+
void CRow::operator=(const double &array[])
  {
   int size=ArraySize(array);
   if(size==0) return;
//--- remplissons le tableau par les valeurs
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| L'opération de l'affectation pour CRow                           |
//+------------------------------------------------------------------+
void CRow::operator=(const CRow  &r)
  {
   int size=r.Size();
   if(size==0) return;
//--- remplissons le tableau par les valeurs
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=r[i];
//---
  }
//+------------------------------------------------------------------+
//| L'opérateur de la multiplication sur une autre chaîne            |
//+------------------------------------------------------------------+
double CRow::operator*(const CRow &o)
  {
   double res=0;
//--- de la vérification
   int size=Size();
   if(size!=o.Size() || size==0)
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       Print(__FUNCSIG__,": L'erreur de la multiplication de deux matrices, ne coïncide
       return(res);
      }
//--- multiplions les tableaux par élément et sommons les produits
   for(int i=0;i<size;i++)
      res+=m_array[i]*o[i];
//--- le résultat
   return(res);
  }
//+------------------------------------------------------------------+
//| Rend la représentation formatée de la chaîne                     |
//+------------------------------------------------------------------+
string CRow::String(void) const
  {
   string out="";
//--- si la taille du tableau est plus grand que zéro
   int size=ArraySize(m_array);
//---travaillons seulement à la quantité non nulle d'éléments dans le tableau
   if(size>0)
     {
      out="{";
      for(int i=0;i<size;i++)
        {
         //--- rassemblons les valeurs dans une chaîne
         out+=StringFormat(" %G;",m_array[i]);
        }
      out+=" }";
     }
//--- le résultat
   return(out);
  }
//+------------------------------------------------------------------+
//| La classe "La matrice"                                           |
//+------------------------------------------------------------------+
class CMatrix
  {
private:
   CRow              m_rows[];

public:
   //--- les constructeurs et le destructeur
                     CMatrix(void);
                     CMatrix(int rows)  { ArrayResize(m_rows,rows);            }
                    ~CMatrix(void){};
   //--- la réception de la taille de la matrice
   int               Rows()       const { return(ArraySize(m_rows));           }
   int               Cols()       const { return(Rows()>0? m_rows[0].Size():0); }
   //---rend les valeurs de la colonne en forme de la chaîne CRow
   CRow              GetColumnAsRow(const int col_index) const;
   //---rend la chaîne avec les valeurs de la matrice
   string            String(void) const;
   //--- l'opérateur de l'indexation rend la chaîne selon son numéro
   CRow *operator[](int i) const        { return(GetPointer(m_rows[i]));       }
   //--- l'opérateur de l'addition
   CMatrix           operator+(const CMatrix &m);
   //--- l'opérateur de la multiplication
   CMatrix           operator*(const CMatrix &m);
   //---l'opérateur de l'affectation
   CMatrix          *operator=(const CMatrix &m);
  };
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Le constructeur par défaut crée un tableau de chaînes de taille zéro |
//+------------------------------------------------------------------+
CMatrix::CMatrix(void)
  {
//--- la quantité nulle de chaînes dans la matrice
   ArrayResize(m_rows,0);
//---
  }
//+------------------------------------------------------------------+
//| Rend les valeurs de la colonne en forme de la chaîne CRow        |
//+------------------------------------------------------------------+
CRow  CMatrix::GetColumnAsRow(const int col_index) const
  {
//--- la variable pour la réception des valeurs d'une colonne
   CRow row();
//---le nombre de chaînes dans la matrice
   int rows=Rows();
//--- si le nombre de chaînes est plus grande que zéro, éxécutons l'opération
   if(rows>0)
     {
      //--- le tableau pour la réception des valeurs avec l'index col_index
      double array[];
      ArrayResize(array,rows);
      //--- le remplissage d'un tableau
      for(int i=0;i<rows;i++)
        {
         //--- la vérification du numéro de la colonne pour l' i-ème chaîne pour la sc
         if(col_index>=this[i].Size())
           {
            Print(__FUNCSIG__,": L'erreur! Le numéro de la colonne ",col_index,"> de l
            break; // row restera l'objet non initialisé
           }
         array[i]=this[i][col_index];
        }
      //--- créons la chaîne CRow à la base des valeurs du tableau
      row=array;
     }
//--- le résultat
   return(row);
  }
//+------------------------------------------------------------------+
//| L'addition de deux matrices                                      |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator+(const CMatrix &m)
  {
//--- le nombre de chaînes et de colonnes dans la matrice transmise
   int cols=m.Cols();
   int rows=m.Rows();
//--- la matrice pour la réception du résultat de l'addition
   CMatrix res(rows);
//--- les tailles de la matrice doivent coïncider
   if(cols!=Cols() || rows!=Rows())
     {
      //--- on ne peut pas faire l'addition
      Print(__FUNCSIG__,": L'erreur de l'addition de deux matrices, ne coïncident pas
      return(res);
     }
//--- le tableau auxiliaire
   double arr[];
   ArrayResize(arr,cols);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- trions les chaînes pour l'addition
   for(int i=0;i<rows;i++)
     {
      //--- enregistrons les résultats des additions des chaînes des matrices au table
      for(int k=0;k<cols;k++)
        {
         arr[k]=this[i][k]+m[i][k];
        }
      //--- mettons le tableau dans la chaîne de la matrice
      res[i]=arr;
     }
//---rendons le résultat de l'addition des matrices
   return(res);
  }
//+------------------------------------------------------------------+
//| La multiplication de deux matrices                               |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator*(const CMatrix &m)
  {
//--- le nombre de colonnes de la première matrice le nombre de chaînes de la matrice
   int cols1=Cols();
   int rows2=m.Rows();
   int rows1=Rows();
   int cols2=m.Cols();
//--- la matrice pour la réception du résultat de l'addition
   CMatrix res(rows1);
//---les matrices doivent être coordonnées
   if(cols1!=rows2)
     {
      //--- on ne peut pas produire la multiplication
      Print(__FUNCSIG__,": L'erreur de la multiplication de deux matrices, le format r
            "- le nombre de colonnes dans le premier facteur doit être égal au nombre
      return(res);
     }
//---le tableau auxiliaire
   double arr[];
   ArrayResize(arr,cols1);
//---remplissons la chaîne dans la matrice de produit
   for(int i=0;i<rows1;i++)// trions les chaînes
     {
      //--- remettons à zéro le tableau-récepteur
      ArrayInitialize(arr,0);
      //--- trions les éléments dans la chaîne
      for(int k=0;k<cols1;k++)
        {
         //--- prenons de la matrice m les valeurs de la k-ème colonne en forme de la
         CRow column=m.GetColumnAsRow(k);
         //--- multiplions deux chaînes et enregistrons le résultat de la multiplicati
         arr[k]=this[i]*column;
        }
      //--- plaçons le tableau arr[] à la i-ème chaîne de la matrice
      res[i]=arr;
     }
//--- rendons le produit de deux matrices
   return(res);
  }
//+------------------------------------------------------------------+
//| L'opération de l'affectation                                     |
//+------------------------------------------------------------------+
CMatrix *CMatrix::operator=(const CMatrix &m)
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- apprenons et spécifions la quantité de chaînes
   int rows=m.Rows();
   ArrayResize(m_rows,rows);
//--- remplissons nos chaînes par les valeurs des chaînes de la matrice transmise
   for(int i=0;i<rows;i++) this[i]=m[i];
//---
   return(GetPointer(this));
  }
//+------------------------------------------------------------------+
//| La représentation de chaîne de la matrice                        |
//+------------------------------------------------------------------+
string CMatrix::String(void) const
  {
   string out="";
   int rows=Rows();
//--- formons par les chaînes
   for(int i=0;i<rows;i++)
     {
      out=out+this[i].String()+"\r\n";
     }
//--- le résultat
   return(out);
  }
```

Voir aussi

La surcharge, Les opérations arithmétiques, La surcharge des fonctions, Les priorités et l'ordre des opérations

© 2000-2015, MetaQuotes Software Corp.

La description des fonctions extérieures

Les fonctions externes définies dans un autre module doivent être décrites explicitement. La description inclut le type de retour, le nom de la fonction et les paramètres d'entrées avec leurs types. L'absence d'une telle description peut conduire à des erreurs lors de la compilation, la construction ou l'exécution d'un programme. Pendant la description d'un objet extérieur, utilisez le mot-clé *#import* indiquant le module.

Exemples:

```
#import "user32.dll"
  int     MessageBoxW(int hWnd ,string szText,string szCaption,int nType);
  int     SendMessageW(int hWnd,int Msg,int wParam,int lParam);
#import "lib.ex5"
  double  round(double value);
#import
```

A l'aide d'importation, il est facile de décrire des fonctions qui sont appelées de DLL externe ou des bibliothèques compilées EX5. Les bibliothèques EX5 sont les fichiers compilés ex5, qui ont la propriété library. On peut importer des bibliothèques EX5 seulement les fonctions décrites par le modificateur export.

À l'utilisation partagée des bibliothèques DLL et EX5 il faut se rappeler ce que les bibliothèques doivent avoir les noms différents (en dehors de la dépendance des répertoires de leur placement). Toutes les fonctions importées reçoivent le domaine de la visibilité "le nom du fichier" de la bibliothèque.

Exemple:

```
#import "kernel32.dll"
   int GetLastError();
#import "lib.ex5"
   int GetLastError();
#import

class CFoo
  {
public:
   int            GetLastError() { return(12345); }
   void           func()
     {
      Print(GetLastError());           // l'appel de la méthode de la classe
      Print(::GetLastError());         // l'appel de la fonction MQL5
      Print(kernel32::GetLastError()); // l'appel de la fonction kernel32.dll
      Print(lib::GetLastError());      // l'appel de la fonction lib.ex5
     }
  };

void OnStart()
  {
   CFoo foo;
   foo.func();
```

© 2000-2015, MetaQuotes Software Corp.

```
}
```

Voir aussi

Surcharge, Fonctions virtuelles, Polymorphisme

© 2000-2015, MetaQuotes Software Corp.

L'exportation des fonctions

Il est possible d'utiliser dans le programme mql5 la fonction déclarée dans un autre programme mql5 avec le postmodificateur *export*. Cette fonction est exportée, et elle est disponible pour être appelée à partir d'autres programmes après la compilation.

```
int Function() export
  {
  }
```

Ce modificateur indique au compilateur d'entrer la fonction dans le tableau des fonctions EX5, exportées par ce fichier ex5 exécutable. Seulement les fonctions avec un tel modificateur sont disponibles («visibles») des autres programmes mql5.

La propriété library indique au compilateur que ce fichier EX5 sera une bibliothèque, et le compilateur marquera ça dans le titre EX5.

Toutes les fonctions qui sont prévues comme exportées, doivent être marqués par le modificateur *export*.

Voir aussi

Surcharge, Fonctions virtuelles, Polymorphisme

© 2000-2015, MetaQuotes Software Corp.

Les fonctions de traitement des événements

Le langage MQL5 fournit le traitement de quelques événeménts prédéterminés. Les fonctions pour le traitement de ces événements doivent être définies dans le programme MQL5; le nom de fonction, le type de la valeur de retour, la composition de paramètres (s'ils existent) et leurs types doivent être strictement conformer à la description de la fonction qui traite l'événement.

C'est le type de valeur de retour et le type des paramètres du gestionnaire d'événements du terminal client identifie la fonction qui traite tel ou tel autre événement. Si les autres paramètres qui ne satisfont pas aux descriptions suivantes, énumérées dans la fonction correspondante ou un autre type de la valeur de retour est indiqué alors cette fonction ne sera pas utilisé pour le traitement des événements.

OnStart

La fonction OnStart() est le gestionnaire d'événement Start, qui est automatiquement généré **seulement** pour les scripts démarrés à l'exécution. Elle doit avoir le type void, elle n' a pas de paramètres:

```
void OnStart();
```

Il est possible de spécifier le type de la valeur de retour int pour la fonction OnStart().

OnInit

La fonction OnInit() est le gestionnaire d'événement Init. Elle peut avoir le type void ou int, elle n' a pas de paramètres:

```
void OnInit();
```

L'événement Init est généré immédiatement après le téléchargement d'un expert ou d'un indicateur, pour les scripts de cet événement ne génèré pas. La fonction OnInit() est utilisé pour l' initialisation. Si OnInit() a type de la valeur de retour int, un code non zéro de retour indique une initialisation échouée et génère l'événement Deinit avec le code de raison de la déinitialisation REASON_INITFAILED.

Pour l'optimisation des paramètres d'entrée de l'expert il est recommandé d'utiliserles valeurs de l'énumération ENUM_INIT_RETCODE à titre du code du retour. Ces valeurs sont destinées à l'organisation de la gestion au procès de l'optimisation, y compris pour le choix desdes agents du testles plus convenants. Directement à l'initialisation de l'expert avant du lancement du test on peut demander l'information sur la configuration et les ressources de l'agent (le nombre de noyaux, le volume de la mémoire libre etc.) à l'aide de la fonction TerminalInfoInteger(). Et à la base de l'information reçue permettre l'utilisation de cet agent de test, ou le refuser à l'optimisation de cet expert.

ENUM_INIT_RETCODE

Identificateur	La description
INIT_SUCCEEDED	L'initialisation a passé avec succès, on peut continuer le test de l'expert.

© 2000-2015, MetaQuotes Software Corp.

	Ce code signifie la même chose que la valeur nulle – l'initialisation de l'expert dans le testeur a passé avec succès.
INIT_FAILED	Une mauvaise initialisation, il n'y a pas de sens de continuer le test à cause des erreurs incorrigibles. Par exemple, on n'a pas réussi à créer l'indicateur nécessaire au travail de l'expert. Le retour de cette valeur signifie la même chose que le retour de la valeur différent du zéro – l'initialisation de l'expert dans le testeur n'est pas réussie.
INIT_PARAMETERS_INCORRECT	Est destiné à l'identification par le programmeur de l'ensemble incorrect des paramètres d'entrée, dans un tableau total de l'optimisation la ligne du résultat avec un tel code du retour sera illuminée par la couleur rouge. Le test pour cet ensemble des paramètres de l'expert ne sera pas fait, l'agent est libre pour la réception d'une nouvelle tâche. A la réception de cette valeur le testeur des stratégies ne transmetta pas cette tâche aux autres agents pour l'exécution répétée.
INIT_AGENT_NOT_SUITABLE	Les erreurs dans le travail du programme à l'initialisation ne sont pas apparues, mais pour quelques raisons cet agent ne convient pas pour faire le test. Par exemple, une mémoire insuffisante, il n'y a pas du soutien OpenCL etc. Après le retour de ce code l'agent ne recevra plus les tâches jusqu'à la finde cette optimisation.

La fonction OnInit() du type void toujours dénote l'initialisation réussie.

OnDeinit

La fonction OnDeinit() est appelée en cadres de deinitialisation et elle est un gestionnaire d'événement Deinit. Doit être déclarée avec le type void et avoir un seul paramètre de type const int , qui contient le code de raison de la déinitialisation. Si un autre type est déclaré, le compilateur émet un avertissement, mais la fonction ne sera pas appelée. Pour les scripts l' événement Deinit ne se génère pas et voilà pourquoi on ne peut pas utiliser la fonction OnDeinit() dans les scripts.

```
void OnDeinit(const int reason);
```

L' événement Deinit se génère pour les experts et les indicateurs dans les cas suivants:

- avant la réinitialisation en raison de changements des symboles ou la période de diagramme, à laquelle le programme mql5 est attaché;

© 2000-2015, MetaQuotes Software Corp.

- avant la réinitialisation à cause de changement des paramètres d'entrée;
- avant le décharge du programme mql5.

OnTick

L' événement NewTick se génère **seulement pour les experts** quand on recoit un tick nouveau pour le symbole, dont au diagramme l'expert est attaché. Il est inutile de définir la fonction OnTick() à l'indicateur d'utilisateur ou le script, parce que l'événement Tick ne se génère pas pour eux.

L' événement NewTick se génère seulement pour les experts, mais cela ne signifie pas que les experts doivent avoir une fonction OnTick(), parce que non seulement les événements NewTick se génèrent pour les experts, mais aussi les événements Timer, BookEvent et ChartEvent. Doit être déclaré avec le type void, elle n'a pas de paramètres:

```
void OnTick();
```

OnTimer

La fonction OnTimer() est appelée lorsque l'événement Timer se produit, qui est généré par le minuteur de système seulement pour les experts et pour les indicateurs - elle ne peut pas être utilisé dans les scripts. La fréquence de venue de cet événement s'établit pendant la souscription d'acquisition par la fonction EventSetTimer() des avertissement d' événement Timer.

Le désabonnement de réception des événements de minuteur pour un Expert particulier se fait par la fonction EventKillTimer(). La fonction doit être définie avec le type void, elle n'a pas de paramètres:

```
void OnTimer();
```

Il est recommandé d'appeler une fonction EventSetTimer() une fois dans la fonction OnInit(), et la fonction EventKillTimer() appeller une fois dans OnDeinit().

Chaque expert et chaque indicateur travaille avec son minuteur, et reçoit les événements seulement de lui. A l'achèvement de travail du programme mql5 le minuteur s'est détruit de force s'il a été créé, mais n'était pas désactivé par la fonction EventKillTimer().

OnTrade

La fonction est appelée dans le cadre de la venue Trade, qui se produit quand on change la liste des ordres placés et des positions ouvertes, d'histoire des ordres et d'histoire des transactions. En cadre de n'importe quelle activité commerciale (l'exposition de l'ordre remis, l'ouverture/la clôture de position, l'arrangement des Stops, déclenchement des ordres remis etc) l'histoire d'ordres et des transactions et/ou la liste des positions et des ordres courants se change à la manière appropriée.

```
void OnTrade();
```

L'utilisateur doit vérifier de lui-même un état de compte commercial dans le code à la réception de cet événement (si c'est exigé par les conditions de stratégie commerciale). Si l'appel de fonction OrderSend() a achevé avec succès et a rendu une valeur true - cela signifie que le serveur commercial a mis l'ordre dans la queue pour l'exécution et lui a accordé le numéro de ticket. Aussitôt que le serveur traite cet ordre l'événement Trade sera génerer. Et si l'utilisateur se souvient de la valeur du ticket pendant le traitement de l'événement OnTrade(), il peut savoir de ce ticket ce qui s'est passé avec l'ordre.

© 2000-2015, MetaQuotes Software Corp.

OnTradeTransaction

A la suite de l'exécution des actions définies avec le compte commercial, son état change. Ces actions comprennent:

- L'envoi de la demande commerciale par n'importe quelle application MQL5- dans le terminal de client à l'aide des fonctions OrderSend et OrderSendAsync et son exécution ultérieure;
- L'envoi de la demande commerciale par l'interface graphique du terminal et son exécution ultérieure;
- le fonctionnement des ordres remis et des ordres stop sur le serveur;
- L'exécution des opérations sur le côté du serveur commercial.

A la suite de ces actions, les transactions commerciales sont exécutées pour le compte:

- le traitement de la demande commerciale;
- le changement des ordres ouverts;
- le changement de l'histoire des ordres;
- le changement de l'histoire des marchés;
- le changement des positions.

Par exemple, à l'envoi de l'ordre de marché sur l'achat, il est traité, l'ordre correspondant sur l'achat est créé pour le compte, se passe l'exécution de l'ordre, son suppression de la liste des ouverts, l'ajout à l'histoire des ordres, ensuite le marché correspondant est ajouté à l'histoire et une nouvelle position est créée. Toutes ces actions sont les transactions commerciales. L'arrivée de chaque telle transaction au terminal est l'événement TradeTransaction. Il appelle le gestionnaire OnTradeTransaction

```
void  OnTradeTransaction(
   const MqlTradeTransaction&    trans,        // la structure de la transaction comme
   const MqlTradeRequest&        request,      // la structure de la demande
   const MqlTradeResult&         result        // la structure de la réponse
   );
```

Le gestionnaire contient trois paramètres:

- **trans** -dans ce paramètre est transmise la structure MqlTradeTransaction, décrivant la transaction commerciale, appliquée au compte commercial;
- **request** - dans ce paramètre est transmise la structure MqlTradeRequest, décrivant la demande commerciale;
- **result** -dans ce paramètre est transmise la structure MqlTradeResult, décrivant le résultat de l'exécution de la demande commerciale.

Les deux derniers paramètres**request** et **result**sont remplis par les valeurs seulement pour la transaction du type TRADE_TRANSACTION_REQUEST, on peut recevoir l'information sur la transaction du paramètre *type* de la variable**trans**. Prêtez attention que dans ce cas le champ*request_id* dans la variable**result** contient l'identificateur de la demande commerciale **request**, à la suite de l'exécution de qui était produite la transaction commerciale, décrite dans la variable**trans**. La présence de l'identificateur de la demande permet de lier l'action éxécutée (l'appel des fonctions OrderSend ou OrderSendAsync) à la suite de cette action transmise dansOnTradeTransaction().

Une demande commerciale envoyée du terminal manuellement ou par les fonctions commerciales OrderSend()/OrderSendAsync(), peut générer quelques transactions commerciales consistants sur le serveur commerciale. Dans ce cas, l'ordre de l'arrivée de ces transactions dans le terminal n'est pas garanti, c'est pourquoi on ne peut pas construire son propre algorithme commercial sur l'attente de

© 2000-2015, MetaQuotes Software Corp.

l'entrée de certaines transactions commerciales après l'arrivée des autres. En outre, les transactions peuvent se perdre à livraison du serveur vers le terminal.

- Tous les types des transactions commerciales sont décrits dans l'énumérationENUM_TRADE_TRANSACTION_TYPE.
- La structure MqlTradeTransaction, décrivante la transaction commerciale, est remplie différemment en fonction du type de la transaction commerciale . Par exemple pour les transaction de ce type TRADE_TRANSACTION_REQUEST il est nécessaire d'analyser seulement un champ - type (le type de la transaction commerciale). Pour la réception de l'information supplémentaire il est nécessaire d'analyser les deuxièmes et troisièmes paramètres de la fonction OnTradeTransaction (request et result). L'information détaillée se trouve dans le paragraphe "La structure de la transaction commerciale".
- Pas toute l'information accessible sur les ordres et les marchés et les positions (par exemple, le commentaire) est transmise dans la description de la transaction commerciale. Pour la réception de l'information élargie il faut utiliser les fonctions OrderGet*, HistoryOrderGet*, HistoryDealGet* et PositionGet*.

Après l'application des transactions commerciales vers le compte de client, ils se placent successivement au tour des transactions commerciales du terminal, d'où elles sont déjà transmises successivement au point de l'entrée OnTradeTransaction dans l'ordre de l'entrée au terminal.

Pendant le traitement des transactions commerciales par l'expert à l'aide d'un gestionnaire OnTradeTransaction, le terminal continue à traiter des nouvelles transactions commerciales entrantes. Ainsi, l'état du compte commercial peut changer déjà en train du travail OnTradeTransaction. Par exemple, quand le programme MQL5 traite l'événement de l'ajout du nouvel ordre, il peut être exécuté, supprimé de la liste des ouverts et déplacé à l'histoire. Par la suite le programme sera informé de tous ces événements.

La longueur du tour des transactions est 1024 éléments. Si OnTradeTransaction traitera la transaction suivante pendant trop longtemps, les vieilles transactions dans le tour peuvent être remplacées par les nouvelles.

- En général, il n'y a pas de rapport exact selon le nombre des appels OnTrade et OnTradeTransaction. Un appel OnTrade correspond à un ou quelques appels OnTradeTransaction.
- OnTrade est appelé après les appels correspondants OnTradeTransaction.

OnTester

La fonction OnTester() est le gestionnaire de l'événement Tester, qui est généré automatiquement à la fin de test historique de l'expert dans l'intervalle spécifié des dates. La fonction doit être définie avec le type double, elle n'a pas de paramètres:

```
double OnTester();
```

La fonction est appelée directement devant l'appel de la fonction OnDeinit() et a le type de la valeur rendue double. La fonction OnTester() peut être utilisée seulement dans les experts au test et est destinée en premier lieu pour le calcul d'une certaine valeur utilisée à titre du critère Custom maxà l'optimisation génétique des paramètres d'entrée.

À l'optimisation génétique le triage des résultats dans la limite d'une génération est produit selon la

© 2000-2015, MetaQuotes Software Corp.

diminution. C'est-à-dire, du point de vue du critère de l'optimisation les meilleurs les résultats avec la plus grande valeur sont considérés (pour le critère de l'optimisation Custom max on a égard les valeurs, rendues par la fonction OnTester). Les valeurs pires à un tel triage se placent à la fin et sont rejetés par la suite et ne prennent pas part à la formation de la génération suivante.

OnTesterInit

La fonction OnTesterInit() est le gestionnaire de l'événementTesterInit, qui est généré automatiquement avant l'optimisation de l'expert dans le testeur des stratégies. La fonction doit être définie avec le type void, elle n'a pas des paramètres:

```
void OnTesterInit();
```

L'expert qui a le gestionnaire OnTesterDeinit () ou OnTesterPass (), au lancement de l'optimisation est chargé automatiquement sur le graphique séparé du terminal avec le symbole indiqué dans le testeur et la période, et reçoit l'événement TesterInit. La fonction est destinée à l'initialisation de l'expert avant l'optimisation pourle traitement suivant des résultats de l'optimisation.

OnTesterPass

La fonction OnTesterPass() est le gestionnaire de l'événement TesterPass, qui est généré automatiquement à l'entrée de la trame pendant l'optimisation de l'expert dans le testeur des stratégies. La fonction doit être définie avec le type void, elle n'a pas des paramètres:

```
void OnTesterPass();
```

L'expert qui a le gestionnaire OnTesterPass() est chargé automatiquement sur le graphique séparé du terminal avec le symbole/période indiqué pour le test et reçoit les événements TesterPass pendant l'optimisation à l'entrée de la trame. La fonction est destinée au traitement dynamique des résultats de l'optimisation toute de suite, sans attendre son achèvement. L'ajout des trames est faite par la fonction FrameAdd(), qui peut être appelée après un seul passage dans le gestionnaire OnTester().

OnTesterDeinit

La fonction OnTesterDeinit() est le gestionnaire de l'événement TesterDeinit, qui est généré automatiquement après l'achèvement de l'optimisation de l'expert dans le testeur des stratégies. La fonction doit être définie avec le type void, elle n'a pas des paramètres:

```
void OnTesterDeinit();
```

L'expert qui a le gestionnaire TesterDeinit() est chargé automatiquement sur le graphique au lancement de l'optimisation et reçoit l'événement TesterDeinit après son achèvement. La fonction est destinée au traitement final de tous les résultats de l'optimisation.

OnBookEvent

La fonction OnBookEvent() est un gestionnaire d'événement BookEvent. L'événement BookEvent se génère seulement pour les experts et pour les indicateurs quand l'état de la profondeur de marché se change (Depth of Market). Elle doit avoir le type void et un paramètre de type string:

```
void OnBookEvent (const string& symbol);
```

Pour recevoir des événements BookEvent pour n'importe quel symbole, il faut juste préventivement de s'abonner pour recevoir ces événements pour ce symbole à l'aide de la fonction MarketBookAdd(). Pour

© 2000-2015, MetaQuotes Software Corp.

désabonner de recevoir des événements BookEvent pour un symbole particulier, il faut appeler la fonction MarketBookRelease().

A la différence d'autres événements, l'événement BookEvent est diffusé. Cela signifie que si un expert souscrit à la réception des événements de BookEvent à l'aide de la fonction MarketBookAdd, tous les autres experts qui ont un gestionnaire OnBookEvent(), recevront cet événement. Il est donc nécessaire d'analyser le nom du symbole, qui est transmis au gestionnaire en tant que le paramètre *const string& symbol*.

OnChartEvent

OnChartEvent() est le gestionnaire de groupe des événements ChartEvent:

- CHARTEVENT_KEYDOWN — l'événement de la pression du clavier, quand la fenêtre du graphique se trouve dans le tour;
- CHARTEVENT_MOUSE_MOVE — les événements du déplacement de la souris et la pression des boutons de la souris (si la propriété est définie pour le graphique CHART_EVENT_MOUSE_MOVE=true);
- CHARTEVENT_OBJECT_CREATE — l'événement de création d'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_CREATE=true);
- CHARTEVENT_OBJECT_CHANGE — l'événement du changement des propriétés de l'objet par le dialogue des propriétés;
- CHARTEVENT_OBJECT_DELETE — l'événement de la suppression de l'objet graphique si la propriété est définie pour le graphique CHART_EVENT_OBJECT_DELETE=true);
- CHARTEVENT_OBJECT_CLICK — l'événement de clic de souris sur l'objet graphique, appartenant au graphique;
- CHARTEVENT_OBJECT_DRAG — l'événement d'un mouvement d'objet graphique en utilisant la souris;
- CHARTEVENT_OBJECT_ENDEDIT — l'événement de la fin de l'édition du texte dans le champ d'entrée d'objet graphique LabelEdit;
- CHARTEVENT_CHART_CHANGE — l'événement du changement du graphique;
- CHARTEVENT_CUSTOM+n — l'identificateur d'événement d'utilisateur, où n se trouve dans la gamme de 0 à 65535.
- CHARTEVENT_CUSTOM_LAST — le dernier identificateur admissible d'événement d'utilisateur (CHARTEVENT_CUSTOM+65535).

La fonction peut être appelée dans les experts et les indicateurs, elle doit avoir le type void et 4 paramètres:

```
void OnChartEvent(const int id,         // identificateur de l'événement
                  const long& lparam,   // paramètre d'événement de type long
                  const double& dparam, // paramètre d'événement de type double
                  const string& sparam  // paramètre d'événement de type string
  );
```

Pour chaque type de l'événement les paramètres d'entrée de la fonction OnChartEvent () ont les valeurs définies, qui sont nécessaires au traitement de cet événement. Dans le tableau on énumère les événements et les valeurs, qui sont transmises par les paramètres.

Evénement	La valeur du	La valeur du	La valeur du	La valeur du

© 2000-2015, MetaQuotes Software Corp.

	paramètre id	paramètre lparam	paramètre dparam	paramètre sparam
L'événement de la pression du clavier	CHARTEVENT_KEYDOWN	le code de la touche appuyée	Le nombre de pressions du bouton générées au cours de sa rétention dans l'état appuyé	La valeur de ligne d'un masque de bit, décrivant le statut des boutons du clavier
Les événements de la souris(si la propriété est définie pour le graphique CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	La coordonnée X	La coordonnée Y	La valeur de chaîne d'un masque de bits décrivant l'état de boutons de souris
L'événement de la création de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_CREATE=true)	CHARTEVENT_OBJECT_CREATE	—	—	Le nom de l'objet graphique créé
L'événement du changement des propriétés de l'objet par le dialogue des propriétés	CHARTEVENT_OBJECT_CHANGE	—	—	Le nom de l'objet changé graphique
L'événement de la suppression de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_DELETE=true)	CHARTEVENT_OBJECT_DELETE	—	—	Le nom de l'objet graphique supprimé
L'événement du clic de la souris sur le graphique	CHARTEVENT_CLICK	la coordonnée X	la coordonnée Y	—
L'événement du clic de la souris	CHARTEVENT_OBJECT_CLICK	la coordonnée X	la coordonnée Y	Le nom de l'objet

© 2000-2015, MetaQuotes Software Corp.

sur le graphique				graphique, sur lequel l'événement a eu lieu
L'événement du déplacement de l'objet graphique à l'aide de la souris	CHARTEVENT_OBJECT_DRAG	—	—	Le nom de l'objet graphique déplacé
l'événement de la fin de l'édition du texte dans le champ d'entrée de l'objet graphique "Le champ de l'entrée"	CHARTEVENT_OBJECT_ENDEDIT	—	—	Le nom de l'objet graphique "Le champ de l'entrée", où l'édition du texte a fini
L'événement du changement du graphique	CHARTEVENT_CHART_CHANGE	—	—	—
L'événement d'utilisateur avec le numéro N	CHARTEVENT_CUSTOM+N	La valeur spécifiée par la fonctionEventChartCustom()	La valeur spécifiée par la fonctionEventChartCustom()	La valeur spécifiée par la fonctionEventChartCustom()

OnCalculate

La fonction OnCalculate() est appelée seulement dans les identificateur d'utilisateurs quand il est nécessaire de calculer les valeurs des indicateurs par l'événement Calculate. Cela se produit généralement quand un nouveau tick sur le symbole est reçue pour laquelle l'indicateur est calculé. En même temps l'indicateur n'est pas tenu d'être attaché à aucun graphique des prix de ce symbole.

La fonction OnCalculate() doit avoir le type de la valeur de retour int. Deux variantes de définition existent. Dans les limites d'un indicateur il est impossible d'utiliser les deux variantes de la fonction.

La première forme d'appel est destinée à ces indicateurs qui peuvent être calculés dans un tampon des données. L'exemple d'un tel indicateur - Custom Moving Average.

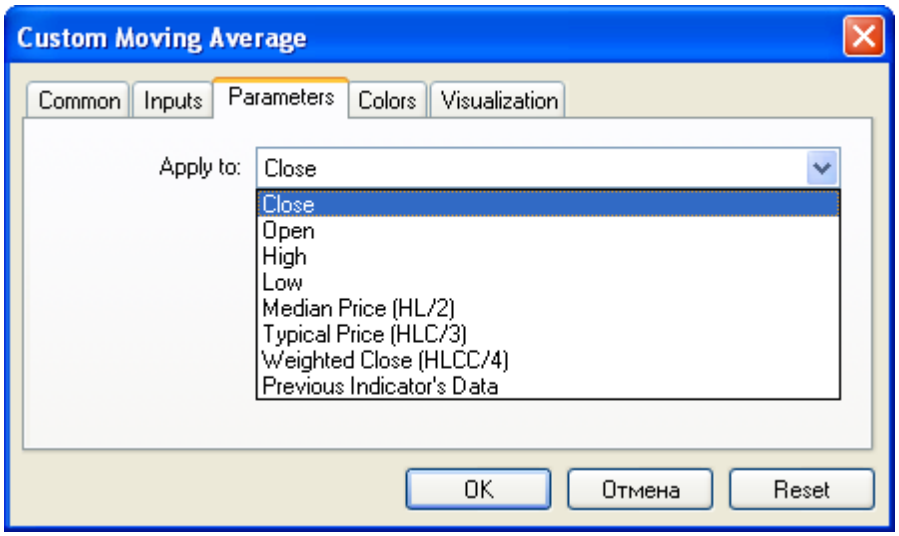
```
int OnCalculate (const int rates_total,      // grandeur de tableau price[]
                 const int prev_calculated,  //Les barres traitées sur un appel précéd
                 const int begin,            // d'où des données significatives commen
                 const double& price[]       // tableau pour le calcul
   );
```

Comme un tableau price[] peut être transmis une des séries temporelles de prix ou un tampon calculé d'un indicateur. Pour déterminer la direction de l'indexation dans le tableau price[], il faut appeler la fonction ArrayGetAsSeries(). Pour ne pas dépendre des défauts, il est nécessaire d'appeler la fonction ArraySetAsSeries() pour ces tableaux, avec lesquels on suppose de travailler.

Le choix de timeserie nécessaire ou de l'indicateur comme le tableau price[] est fait par l'utilisateur

© 2000-2015, MetaQuotes Software Corp.

quand on lance l'indicateur dans l'onglet "Parameters". Pour faire ça il faut indiquer un élément nécessaire dans une liste déroulante de champs "Apply to".



Pour recevoir des valeurs d'un indicateur d'utilisateur des autres programmes mql5 on utilise la fonction iCustom(), qui rend le handle de l'indicateur pour les opérations ultérieures. En plus on peut indiquer aussi le tableau nécessaire price[] ou le handle de l'indicateur. Ce paramètre doit être transmis le dernier dans la liste des variables d'entrée de l'indicateur d'utilisateur.

Exemple:

```
void OnStart()
  {
//---
   string terminal_path=TerminalInfoString(STATUS_TERMINAL_PATH);
   int handle_customMA=iCustom(Symbol(),PERIOD_CURRENT, "Custom Moving Average",13,0,
   if(handle_customMA>0)
      Print("handle_customMA = ",handle_customMA);
   else
      Print("Cannot open or not EX5 file '"+terminal_path+"\\MQL5\\Indicators\\"+"Cust
  }
```

Dans cet exemple la valeur PRICE_TYPICAL (de l'énumération ENUM_APPLIED_PRICE) a été transmis par le dernier paramètre, qui indique que l'indicateur d'utilisateur sera construit sur les prix typiques, reçus comme (High+Low+Close)/3. Si ce paramètre n'est pas spécifié, l'indicateur est basé sur les valeurs PRICE_CLOSE, le cours en clôture de chaque barre.

Un autre exemple représentant la transmission du handle de l'indicateur par le dernier paramètre pour l'indication du tableau price[], est cité dans la description de la fonction iCustom().

Une deuxième forme d'appel est destiné pour tous les autres indicateurs, dans lequel plus qu'une série temporelle est utilisé pour le calcul.

© 2000-2015, MetaQuotes Software Corp.

```
int OnCalculate (const int rates_total,      // valeur des séries temporelles d'entrée
                 const int prev_calculated,  // les barres traitées dans appel précédé
                 const datetime& time[],     // Time
                 const double& open[],       // Open
                 const double& high[],       // High
                 const double& low[],        // Low
                 const double& close[],      // Close
                 const long& tick_volume[],  // Tick Volume
                 const long& volume[],       // Real Volume
                 const int& spread[]         // Spread
   );
```

Les paramètres open[], high[], low[] et close[] contiennent les tableaux avec les prix de l'ouverture, les prix maximal, minimal et les prix de clôture de TIMEFRAME actuel. Le paramètre time[] contient le tableau avec les valeurs du temps de l'ouverture, un paramètre spread[] - le tableau qui contient l'histoire des spreads (si le spread est prévue pour cet instrument commercial). Les paramètres volume[] et tick_volume[] contient conformément l'histoire de volume de la dimension commerciale et de tick.

Pour déterminer la direction de l'indexation dans des tableaux time[], open[], high[], low[], close[], tick_volume[], volume[] et spread[], il faut appeler la fonction ArrayGetAsSeries(). Pour ne pas dépendre des défauts, il faut sûrement appeler une fonction ArraySetAsSeries() pour ces tableaux, avec lesquels on suppose de travailler.

Le premier paramètre rates_total contient le nombre de barres, disponibles pour l'indicateur pour le calcul et correspond au nombre de barres disponibles dans le graphique.

Il faut noter le rapport entre la valeur rendue par la fonction OnCalculate() et le deuxième paramètre d'entrée prev_calculated. Le paramètre prev_calculated pendant l'appel de fonction contient la valeur **retournée** par la fonction OnCalculate() sur un appel **précédent**. Cela permet de réaliser des algorithmes économiques de calcul de l'indicateur d'utilisateur pour éviter des calculs répétés pour les barres, qui n'ont pas changé depuis le précédent lancement de cette fonction.

Pour cela, il est d'habitude suffisant de rendre la valeur du paramètre rates_total, qui contient un certain nombre de barres dans l'appel de la fonction courante. Si depuis le dernier appel de la fonction OnCalculate() les données de prix ont été changé (une histoire plus profonde est téléchargée ou les blancs d'histoire ont été remplis), la valeur du paramètre d'entrée prev_calculated sera mis à la valeur nulle par le terminal.

Note: si la fonction OnCalculate rend la valeur nulle, donc dans la fenêtre DataWindow de terminal de client les valeurs d'indicateur ne se montre pas.

Pour le comprendre mieux serait utile de démarrer l'indicateur dont le code qui est jointe ci-dessous.

Exemple d'indicateur:

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Line
#property indicator_label1  "Line"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDarkBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double         LineBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime& time[],
                const double& open[],
                const double& high[],
                const double& low[],
                const double& close[],
                const long& tick_volume[],
                const long& volume[],
                const int& spread[])
  {
//--- recevons le nombre de barres disponibles pour le symbole actuel et la période su
   int barres=Barres(Symbol(),0);
   Print("Barres = ",barres,", rates_total = ",rates_total,", prev_calculated = ",prev
   Print("time[0] = ",time[0]," time[rates_total-1] = ",time[rates_total-1]);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Voir aussi

Exécution des programmes, Evénement du terminal de client, Travail avec des événements

© 2000-2015, MetaQuotes Software Corp.

Les variables

Déclaration des variables

Les variables doivent être déclarées avant de les utiliser. Pour identifier les variables on utilise des noms uniques. Les descriptions des variables sont utilisées pour leur détermination et pour déclaration des types. La description n'est pas un opérateur.

Les types simples sont:

- char, short, int, long, uchar, ushort, uint, ulong - les nombres entiers;
- color - un nombre entier, représentant la couleur RGB;
- datetime - la date et l'heure, un entier non signé contenant le nombre de secondes depuis 0 heures du 1er Janvier, 1970;
- bool - valeurs logiques true et false;
- double - nombre de double-précision avec une virgule flottante;
- float - nombre de simple précision avec une virgule flottante;
- string - chaînes de caractères.

Exemples:

```
string szInfoBox;
int    nOrders;
double dSymbolPrice;
bool   bLog;
datetime tBegin_Data   = D'2004.01.01 00:00';
color    cModify_Color = C'0x44,0xB9,0xE6';
```

Types complexes et composés:

Les structures - ce sont les types composés de données, construits en utilisant les autres types.

```
struct MyTime
  {
   int hour;    // 0-23
   int minute;  // 0-59
   int second;  // 0-59
  };
...
MyTime strTime; // variable du type de la structure auparavant déclarée MyTime
```

Tant que la structure n'a pas été déclaré, on ne peut déclarer des variables du type de la structure.

Tableaux

Le tableau est l'ordre indexé de données du type identique:

```
int    a[50];       // tableau unidimensionnel de 50 nombres entiers.
double m[7][50];    // tableau bidimensionnel de sept tableaux,
                    // dont chacune se compose de 50 nombres.
```

© 2000-2015, MetaQuotes Software Corp.

```
MyTime t[100];      // tableau qui contient les éléments du type MyTime
```

Seulement un nombre entier peut être un index de tableau. Pas plus de quatredimensionel tableaux sont admis. La numération des éléments du tableau commence à 0. Le dernier élément d'un tableau unidimensionnel a le numéro 1 de moins que la valeur du tableau, cela signifie que l'appel au dernier élément d'un tableau consistant de 50 nombres entiers apparaîtra comme a[49]. Le même concerne des tableaux multidimensionnels - une dimension est indexée de 0 à la grandeur de dimension -1. Le dernier élément d'un tableau bidimensionel de l'exemple sera comme m[6][49].

Les tableaux statiques ne peuvent pas être représentés comme les séries temporelles, la fonction ArraySetAsSeries() n'est pas applicable à eux, qui établit l'accès aux éléments de tableau, de la fin de tableau au commencement. S'il faut fournir un accès à un tableau comme dans les séries temporelles, utiliser l'objet de tableau dynamique.

À l'accès au-delà des frontières de tableau un système executif générera l'erreur critique et l'exécution du programme sera arrêtée.

Spécificateurs d'accès

Les spécificateurs d'accès définient au compilateur comment on peut accéder aux variables, les membres de structures ou de classes.

Le spécificateur const déclare une variable comme une constante et ne permet pas de changer cette variable pendant le temps de l'exécution du programme. Une initialisation simple d'une variable est permise en le déclarant.

Exemples

```
int OnCalculate (const int rates_total,      // grandeur de tableau price[]
                 const int prev_calculated,  // Les barres traitées sur un appel préce
                 const int begin,            // d'où des données significatives commer
                 const double& price[]       // tableau pour le calcul
   );
```

Pour accéder aux membres des structures et des classes on utilise les spécificateurs suivants:

- public - il permet un accès illimitée à la variable ou à la méthode de classe;
- protected - il permet un accès des méthodes pour cette classe, ainsi que des méthodes de classes publiquement héritées. Un autre accès est impossible;
- private - il permet un accès aux variables et des méthodes de la classe seulement des méthodes de cette classe.
- virtual - il est appliqué seulement aux méthodes de la classe (mais pas aux méthodes des structures) et indique au compilateur que cette méthode être placée dans le tableau des fonctions virtuelles de la classe.

Classes de stockage

Il existe trois classes de stockage: static, input et extern. Ces modificateurs d'une classe de stockage indiquent clairement au compilateur que les variables correspondantes sont distribuées dans les zones prédéterminées de la mémoire, appelée un pool mondiale. En plus ces modificateurs indiquent au traitement spécial de données des variables.

© 2000-2015, MetaQuotes Software Corp.

Si la variable, déclarée au niveau local, n'est pas statique, alors l'allocation de la mémoire pour une telle variable est fait automatiquement sur la pile de programme. La désallocation de la mémoire réservée pas pour le tableau statique, est produite aussi automatiquement à la sortie au-delà du domaine de la visibilité du bloc, où le tableau est déclaré.

Voir aussi

Types des données, Encapsulation et extensibilité des types, Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets, Les membres statiques de la classe

© 2000-2015, MetaQuotes Software Corp.

Les variables locales

Une variable déclarée à l'intérieur de n'importe quelle fonction est locale. Le secteur de visibilité de variable locale est limité par les limites de fonction, à l'intérieur de laquelle est elle declarée. Une variable locale peut être initialisée à l'aide de n'importe quelle expression. L'initialisation de la variable locale s'est produit chaque fois à l'appel de la fonction appropriée. Les variables locales sont placées dans la zone de mémoire temporaire de fonction correspondant.

Exemple:

```
int somefunc()
  {
   int ret_code=0;
   ...
   return(ret_code);
  }
```

Le champ d'action (ou le secteur de visibilité) de la variable - c'est une partie du programme, dans lequel on peut se référer à la variable. Les variables déclarées dans un bloc (au niveau interne), ont comme un champ d'action un bloc. Le champ d'action un bloc commence par la déclaration de variable et finit par une accolade droite.

Les variables locales déclarées au début de la fonction, ont le champ d'action un bloc ainsi comme les paramètres de la fonction, qui sont des variables locales. N'importe quel bloc peut contenir des déclarations des variables. Si les blocs sont imbriqués et l'identificateur dans un bloc extérieur a le même nom comme l'identificateur dans le bloc intérieur, l'identificateur de bloc extérieur est "invisible" (caché), jusqu'à l'achèvement du travail du bloc interne.

Exemple:

```
void OnStart()
  {
//---
   int i=5;       // variable de la fonction locale
     {
      int i=10;  // variable de la fonction
      Print("dans le bloc i = ",i); // résultat i=10;
     }
   Print("a l'extérieur de bloc i = ",i);  // résultat i=5;
  }
```

Cela signifie que lors de l'exécution de bloc intérieur, il voit la valeur de ses propres identificateurs locaux au lieu de la valeur d'identificateurs avec des noms identiques dans le bloc extérieur.

Exemple:

```
void OnStart()
  {
//---
   int i=5;       // variable de la fonction locale
   for(int i=0;i<3;i++)
      Print("A l'intérieur for i = ",i);
```

© 2000-2015, MetaQuotes Software Corp.

```
    Print("A l'extérieur de bloc i = ",i);
  }
/* Le résultat d'exécution
A l'intérieur for i = 0
A l'intérieur for i = 1
A l'intérieur for i = 2
A l'extérieur de bloc i = 5
*/
```

Les variables locales déclarées comme static, ont le bloc comme le champ d'action, malgré qu'ils existent dès le début d'exécution du programme.

Pile

Le domaine spécial de la mémoire qui s'appelle la pile est attribué pour stocker des variables locales automatiquement créées dans chaque programme MQL5. Une pile est attribué à toutes les fonctions et la taille d'une pile est 256 kb par défaut, on peut diriger la taille de la pile par la directive du compilateur#property stacksize.

Les variablesstatiques locales sont placés au même endroit que les autres variables statiquesglobales, dans un domaine spécial de la mémoire existant séparément de la pile. Les variables dynamiquementcréées utilisent un domaine de la mémoire séparé de la pile.

La place sur la pile est assignée pour les variables intérieures non statiques à chaque appel de la fonction. A la sortie de la fonction la mémoire devient accessible pour l'utilisation réitérée.

Si l'appel de la deuxième fonction est produit de la première fonction, celle-là occupe à son tour dans la pile le volume nécessaire pour ses variables de la mémoire de la pile restée. Ainsi aux appels imbriqués des fonctions sur la pile la mémoire sera occupée successivement pour chaque fonction. Cela peut conduire à l'insuffisance de la mémoire à l'appel suivant de la fonction, cette situation s'appelle le débordement de la pile.

C'est pourquoi pour les grandes données locales il est préférable d'utiliser la mémoire dynamique - à l'entrée dans la fonction il faut allouer la mémoire dans le système pour les besoins locaux (new, ArrayResize()), à la sortie de la fonction faire la désallocation de la mémoire (delete, ArrayFree()).

Voir aussi

Types des données, Encapsulation et extensibilité des types, Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

Les paramètres formels

Les paramètres, transmis dans la fonction, sont locaux. Le bloc de la fonction est le secteur de visibilité. Les paramètres formels doivent différer des noms des variables externes et les variables locales définies dans une fonction. Dans le bloc de fonction les certaines valeurs peuvent être assignées les paramètres formels. Si le paramètre formel est déclarée avec le modificateur const, alors sa valeur ne peut être changée au sein de la fonction.

Exemple:

```
void func(const int & x[], double y, bool z)
  {
   if(y>0.0 && !z)
      Print(x[0]);
   ...
  }
```

Les paramètres formels peuvent être initialisées par les constantes. Dans ce cas, la valeur d'initialisation est celle par défaut. Les paramètres suivants au paramètre initialisé doivent être également initialisés.

Exemple:

```
void func(int x, double y = 0.0, bool z = true)
  {
   ...
  }
```

En appelant une telle fonction, les paramètres initialisés peuvent être omis, ils seront remplacés par les valeur par défaut.

Exemple:

```
func(123, 0.5);
```

Les paramètres des types simples sont transmis par la valeur, c'est-à-dire les changements de la variable locale correspondante de ce type à l'intérieur de la fonction appelée ne sera pas reflétée à la fonction d'appel. Les tableaux de n'importe quel type et les données du type de structure sont toujours transmis par la référence. S'il est nécessaire d'interdire la modification du tableau ou le contenu de structure, les paramètres de ces types doivent être déclarées avec le mot-clé const.

Il existe la possibilité de transmettre les paramètres de simples types par la référence. Dans ce cas, la modification de ces paramètres refléterai sur les variables correspondantes dans la fonction appelée, transmises par la référence. Pour spécifier qu'un paramètre est transmis par la référence, il faut mettre le modificateur & après le type des données.

Exemple:

```
void func(int& x, double& y, double & z[])
  {
   double calculated_tp;
   ...
```

© 2000-2015, MetaQuotes Software Corp.

```
   for(int i=0; i<OrdersTotal(); i++)
     {
      if(i==ArraySize(z))        break;
      if(OrderSelect(i)==false) break;
      z[i]=OrderOpenPrice();
     }
   x=i;
   y=calculated_tp;
  }
```

Les paramètres, transmis par le lien, ne peuvent pas être initialisés par les valeurs par défaut.

On peut passer le maximum 64 paramètres dans une fonction.

Voir aussi

Les variables Input, Types des données, Encapsulation et extensibilité des types, Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

Les variables statiques

La classe de stockage static définit une variable statique. Le modificateur static est indiqué avant le type de données.

Exemple:

```
int somefunc()
  {
   static int flag=10;
   ...
   return(flag);
  }
```

Une variable statique peut être initialisée par la constante correspondant à son type ou par une expression constante, contrairement à une variable locale simple, qui peut être initialisée par n' importe quelle expression.

Les variables statiques existent du moment d'exécution de programme et sont initialisées une seule fois avant l'appel de fonction spécialisée OnInit(). Si les valeurs initiales ne sont pas spécifiées, alors les variables de classe statique de stockage les valeurs nulle initiales.

Les variables locales, déclarées avec le mot clé static conservent leur valeur pendant tout le temps d'existence de la fonction. Avec chaque appel suivant de fonction les telles variables locales contiennent les valeurs qu'ils avaient pendant un appel précédent.

Toutes les variables dans le bloc, à l'exception des paramètres formels de fonctions peuvent être définies comme statiques.

Exemple:

```
int Counter()
  {
   static int count;
   count++;
   if(count%100==0) Print("Fonction Counter a été déjà appelée ",count," un");
   return count;
  }
void OnStart()
  {
//---
   int c=345;
   for(int i=0;i<1000;i++)
     {
      int c=Counter();
     }
   Print("c =",c);
  }
```

Voir aussi

Types des données, Encapsulation et extensibilité des types, Initialisation des variables, Portée de

© 2000-2015, MetaQuotes Software Corp.

visibilité et la durée de vie des variables, Création et destruction des objets, Les membres statiques de la classe

© 2000-2015, MetaQuotes Software Corp.

Les variables globales

Les variables globales sont créées en plaçant leurs déclarations au lieu de la description d'une fonction. Des variables globales sont définies au même niveau que des fonctions, c'est-à-dire, ils ne sont pas locaux dans aucun bloc.

Exemple:

```
int GlobalFlag=10;   // variable globale
int OnStart()
  {
   ...
  }
```

Le secteur de visibilité des variables locales - est tout le programme, les variables globales sont accessibles depuis toutes les fonctions définies dans le programme. Elles sont initialisées au zéro à moins qu'une autre valeur initiale ne soit explicitement définie. Une variable globale peut être initialisée seulement par une expression constante ou constante qui correspond à son type.

Global variables are initialized only once after the program is loaded into the client terminal memory and before the first handling of the Init event. For global variables representing class objects, during their initialization the corresponding constructors are called. In scripts global variables are initialized before handling the Start event.

Note: il ne faut pas confondre les variables, déclarées au niveau global, avec des variables globales du terminal de client, qui sont accessibles par les fonctions GlobalVariable...().

Voir aussi

Types des données, Encapsulation et extensibilité des types, Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

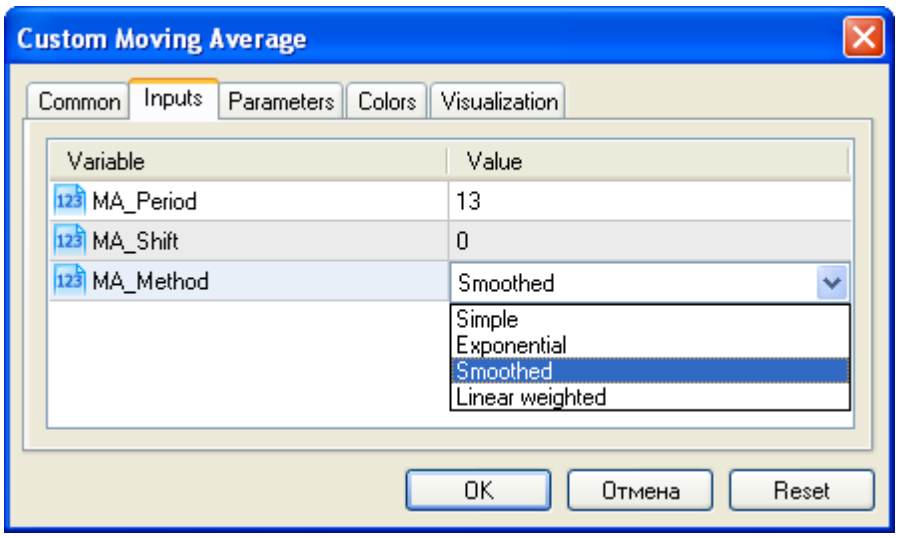
Les variables Input

La classe de stockage input détermine la variable externe. Le modificateur input est indiqué avant le type des données. Il est interdit de changer la valeur d'une variable avec le modificateur input à l'intérieur de programme mql5, telles variables sont accessibles seulement pour la lecture. Seulement un utilisateur peut changer les valeurs des variables input de la fenêtre des propriétés du programme.

Exemple:

```
//--- input parameters
input int             MA_Period=13;
input int             MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMMA;
```

Les variables Input déterminent les paramètres d'entrée du programme, ils sont disponibles de la fenêtre des propriétés du programme.

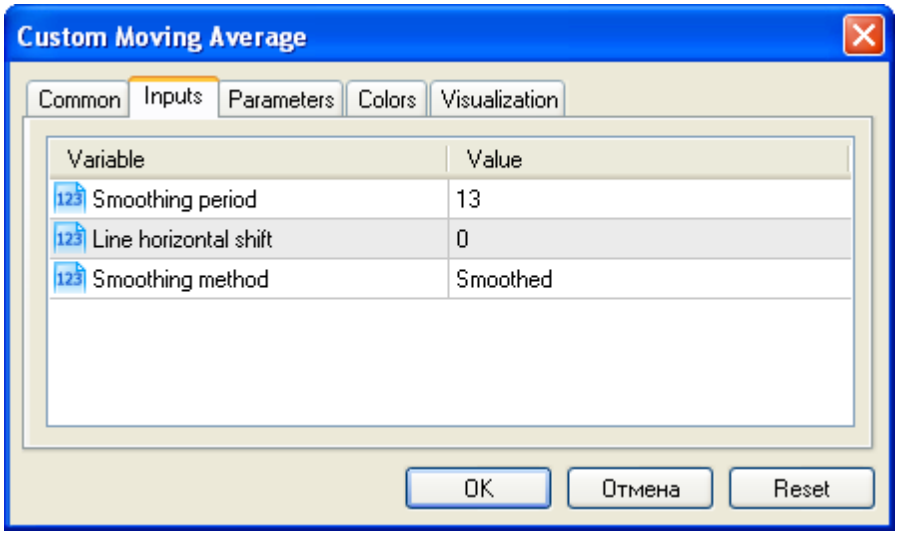


Il est possible de mettre une autre façon de montrer les noms de paramètres d'entrée sur l'onglet "Inputs". Pour cela on utilise le commentaire de chaîne, qui doit être placé après la description du paramètre d'entrée dans la même ligne. Ainsi les noms plus compréhensibles à l'utilisateur font un rapprochement aux paramètres d'entrée.

Exemple:

```
//--- input parameters
input int             InpMAPeriod=13;         // Smoothing period
input int             InpMAShift=0;           // Line horizontal shift
input ENUM_MA_METHOD InpMAMethod=MODE_SMMA;  // Smoothing method
```

© 2000-2015, MetaQuotes Software Corp.



Note: Les tableaux et les variables des types complexes ne peuvent pas être les variables input.

Le passage des paramètres en appelant des indicateurs d'utilisateurs des programmes mql5

Les indicateurs d'utilisateurs sont appelés à l'aide de la fonction iCustom(). Cependant, après le nom de l'indicateur d'utilisateur doivent aller les paramètres en stricte conformité avec la déclaration des variables input de cet indicateur d'utilisateur. Si on indique les paramètres moins que les variables input declarées à l'indicateur d'utilisateur appelé, les paramètres manquants sont remplis des valeurs indiquées pendant la déclaration de variables.

Si on utilise la fonction OnCalculate à l'indicateur d'utilisateur du premier type (c'est-à-dire quand l'indicateur est calculé utilisant le même tableau de données), alors une des valeurs ENUM_APPLIED_PRICE ou le handle d'un autre indicateur doivent être utilisés comme le dernier paramètre en appelant un indicateur d'utilisateur. En même temps tous les paramètres correspondants aux valeurs input doivent être clairement indiqués.

Énumérations comme le paramètre input

Comme les variables input (paramètres d'entrée pour les programmes mql5) on peut utiliser les énumérations intégrées prévues pour la langage MQL5, mais les énumérations, définies par l'utilisateur. Par exemple, nous pouvons créer une énumération dayOfWeek, décrivante les jours de la semaine, et utiliser la variable input pour spécifier une date particulière de la semaine, pas comme une chiffre, mais dans le type plus habituel pour l'utilisateur.

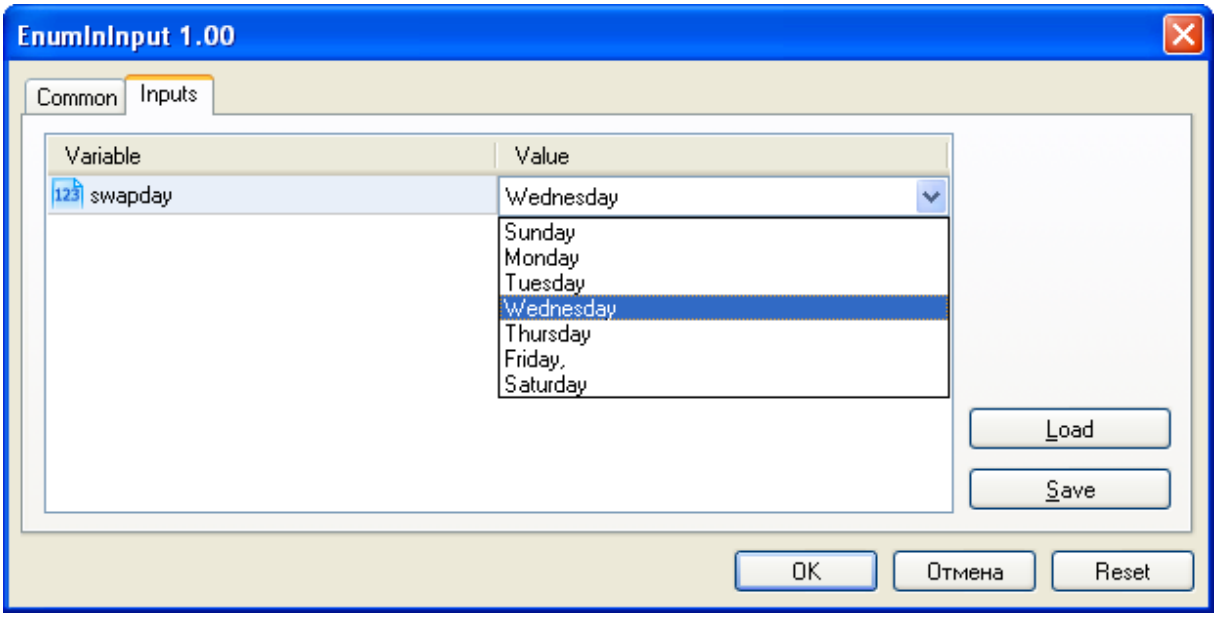
Exemple:

```
#property script_show_inputs
//--- day of week
enum dayOfWeek
  {
   S=0,      // Sunday
   M=1,      // Monday
   T=2,      // Tuesday
   W=3,      // Wednesday
```

© 2000-2015, MetaQuotes Software Corp.

```
   Th=4,     // Thursday
   Fr=5,     // Friday,
   St=6,     // Saturday
  };
//--- input parameters
input dayOfWeek swapday=W;
```

Pour permettre à un utilisateur de choisir une valeur nécessaire parmi la fenêtre de propriétés pendant le lancement du script, nous utilisons l'ordre de préprocesseur #property script_show_inputs. Nous lançons le script à l'exécution et pouvons choisir l'une des valeurs des énumérations dayOfWeek de la liste. Lançons le script EnumInInput et passons à l'onglet "Paramètres". Par défaut, la valeur de paramètre swapday (le jour de charge de swap triple) est l'environnement (W=3), mais nous pouvons spécifier une autre valeur et utiliser cette valeur pour changer le travail du programme.



Le nombre de valeurs possibles de ces énumérations est limité. Pour sélectionner la valeur d'entrée on utilise la liste déroulante. Comme les valeurs affichées dans la liste, on utilise des noms mnémoniques des membres d'énumération. Si un commentaire est associé à un nom mnémonique, comme c'est montré dans cet exemple, on utilise le contenu de commentaire au lieu de nom mnémonique.

Chaque valeur de l'énumération dayOfWeek a sa valeur de 0 à 6, mais dans la liste des paramètres seront montrés les commentaires, spécifiés pour chaque valeur. Cela fournit la flexibilité supplémentaire à écrire des programmes avec les descriptions claires de paramètres d'entrée.

Les variables avec le modificateur sinput

Les variables avec le modificateurinputpermettent non seulement de spécifier les valeurs des paramètres extérieurs au démarrage des programmes, mais aussi jouent un grand rôle à l'optimisation des stratégies commerciales dans le testeur. Chaque variable input déclarée dans l'expert, à l'exception du type string, peut participer à l'optimisation.

Dans certains cas, il peut être nécessaire d'exclure certains paramètres extérieurs du programme de la formation de tous les passages possibles dans le testeur. Pour tels cas, il y a le modificateur de la

© 2000-2015, MetaQuotes Software Corp.

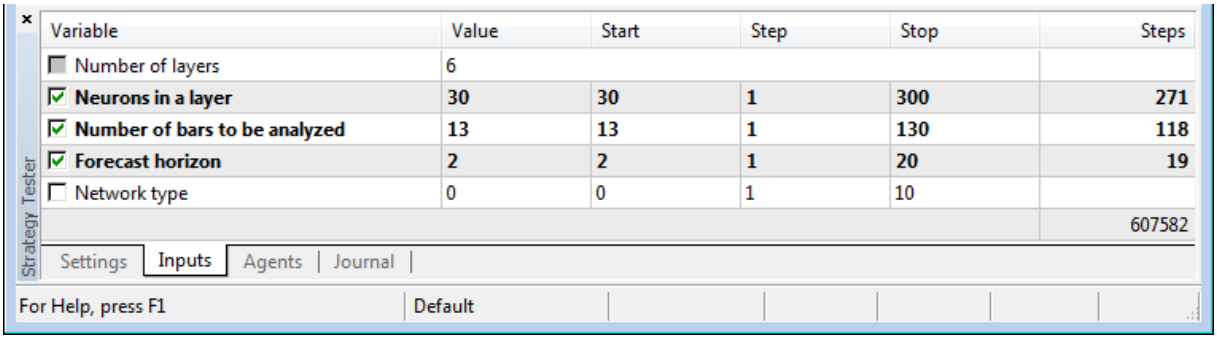
mémoiresinput. sinput - c'est l'orthographe réduit de l'annonce de la variable statique extérieure: sinput = static input. Autrement dit, une telle déclaration dans le code du conseiller

```
sinput       int layers=6;    // Number of layers
```

serait équivalente à une déclaration complète

```
static input int layers=6;    // Number of layers
```

La variable déclarée avec le modificateur sinput est le paramètre d'entrée du programme MQL5, on peut changer la valeur de ce paramètre à son lancement. Mais en cela cette variable ne participe pas au procès de l'optimisation des paramètres d'entrée, c'est-à-dire on ne produit pas le balayage de ses valeurs à la recherche du meilleur ensemble des paramètres selon le critère spécifié.



Sur le dessin est montré que l'expert a 5 paramètres extérieurs, le paramètre "Nombre de couches" est annoncé comme sinput et est égal 6. Ce paramètre ne peut pas changer dans la procédure de l'optimisation de la stratégie commerciale, il faut établir la valeur nécessaire pour lui, qui sera utilisée. Les champs Démarrage, Pas et Stop pour une telle variable ne sont pas accessibles pour l'installation des valeurs.

Ainsi, ayant spécifié le modificateur sinput pour la variable, nous interdisons d'optimiser ce paramètre à l'utilisateur. Cela signifie que dans le testeur des stratégies l'utilisateur ne pourra pas spécifier pour une telle variable les valeurs initiales et finales pour le balayage automatique dans la gamme indiquée pendant l'optimisation.

Mais il y a une exception de la règle donnée – on peut varier les variables sinput dans les tâches de l'optimisation à l'aide de la fonction ParameterSetRange(). Cette fonction est spécialement créée pour la gestion de programme de l'espace des valeurs accessibles pour chaque variable input y compris celles qui sont déclarées commestatic input (sinput). Une autre fonction ParameterGetRange() permet au lancement de l'optimisation (dans le gestionnaire OnTesterInit()) recevoir les valeurs des variables input et en cas de besoin redéfinir le pas du changement et la gamme, dans les limites duquel sera fait le balayage du paramètre optimisé.

Voir aussi

iCustom, Les énumérations, Les propriétes des programmes

© 2000-2015, MetaQuotes Software Corp.

Les variables Extern

Le mot-clé extern est utilisé pour déclarer des identificateurs des variables comme les identificateurs de la classe de stockage statique avec la durée de vie globale. Ces variables existent depuis le début du programme et la mémoire est allouée pour eux et initialisée immédiatement après le début du programme.

Vous pouvez créer des programmes qui se composent de plusieurs fichiers source, pour cela on utilise la directive au préprocesseur #include. Les variables déclarées comme extern avec le même type et l'identificateur ne peut exister dans les différentes fichiers source d'un projet.

En compilant le projet entier, toutes les variables extern avec le même type et un identificateur sont associées à une partie de mémoire de pool global des variables. Les variables Extern sont utiles pour la compilation séparée des fichiers sources. Les variables Extern peuvent être initialisées, mais seulement une fois – l'existence de plusieurs variables extern initialisées du même type et avec le même identificateur il est inacceptable.

Voir aussi

Types des données, Encapsulation et extensibilité des types, Initialisation des variables, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

L'initialisation des variables

Toute variable dans la détermination peut être initialisée. Si une initialisation explicite de variable n'est pas faite, la valeur stockée dans cette variable peut être chacune. Une initialisation implicite n'est pas faite.

Les variables globales et statiques peuvent être initialisées uniquement par une constante du type correspondant ou par expression constante. Les variables locales peuvent être initialisées par toute expression, non seulement par la constante.

L'initialisation des variables globales et statiques est faite une fois. L'initialisation des variables locales est faite chaque fois quand on appelle les fonctions correspondantes.

Exemples:

```
int    n       = 1;
string s       = "hello";
double f[]     = { 0.0, 0.236, 0.382, 0.5, 0.618, 1.0 };
int    a[4][4] = { {1, 1, 1, 1}, {2, 2, 2, 2}, {3, 3, 3, 3}, {4, 4, 4, 4 } };
//--- de tetris
int    right[4]={WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER,
                 WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER};
//--- l'initialisation de tous les champs de la structure par les valeurs nulles
MqlTradeRequest request={0};
```

La liste des valeurs de tableau doit être entourée par des accolades. Les séquences d'initialisation omises sont considérés comme égaux à 0. Au moins une valeur doit être dans la séquence d'initialisation: le premier élément de la structure correspondante ou un tableau sont initialisées par cette valeur, les éléments omis sont considérés comme égaux à zéro.

Si la grandeur du tableau initialisé n'est pas spécifiée, elle est déterminée par le compilateur, basé sur la grandeur de la séquence d'initialisation. Les tableaux multidimensionnels ne peuvent pas être initialisés par la séquence unidimensionnelle (la séquence sans accolades supplémentaires), à part le cas quand un élément initialisant seulement est spécifié (en règle générale, le zéro).

Les tableaux (y compris celles déclarées au niveau local) peuvent être initialisés uniquement par les constantes.

Exemples:

```
struct str3
  {
   int               low_part;
   int               high_part;
  };
struct str10
  {
   str3              s3;
   double            d1[10];
   int               i3;
  };
void OnStart()
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
   str10 s10_1={{1,0},{1.0,2.1,3.2,4.4,5.3,6.1,7.8,8.7,9.2,10.0},100};
   str10 s10_2={{1,0},{0},100};
   str10 s10_3={{1,0},{1.0}};
//---
   Print("1.  s10_1.d1[5] = ",s10_1.d1[5]);
   Print("2.  s10_2.d1[5] = ",s10_2.d1[5]);
   Print("3.  s10_3.d1[5] = ",s10_3.d1[5]);
   Print("4.  s10_3.d1[0] = ",s10_3.d1[0]);
  }
```

L'initialisation partielle est admise pour les variables telles que les structures, la même chose se rapporte et aux tableaux statiques (avec la taille évidemment spécifiée). On peut initialiser un ou quelques premiers éléments d'une structure ou d'un tableau, dans ce cas les éléments restants seront initialisés par les zéros.

Voir aussi

Types des données, Encapsulation et extensibilité des types, Portée de visibilité et la durée de vie des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

La portée de visibilité et la durée de vie des variables

Il existe deux types de portée de visibilité: une portée de visibilité locale et une portée de visibilité globale.

Une variable déclarée en dehors de toutes les fonctions, est placée dans la portée de visibilité globale. L'accès à de telles variables peut être fait de n'importe quel endroit de programme. Ces variables sont situées dans le pool de mémoire, donc leur durée de vie coïncide avec la durée de vie du programme.

Une variable déclarée dans un bloc (la partie de code mise dans des accolades), appartient à la portée locale de visibilité. Une telle variable n'est pas visible (et donc non disponibles) en dehors du bloc dans lequel elle est déclarée. Le cas le plus commun de déclaration locale est une variable déclarée dans une fonction. Une variable déclarée localement, est placée sur la pile et la durée de vie d'une telle variable est égale à la durée de vie de la fonction.

Depuis la portée de visibilité d'une variable locale est le bloc dans lequel elle est déclarée, il est possible de déclarer des variables avec un nom qui correspond aux noms de variables déclarées dans les autres blocs; aussi bien que de ceux déclarés à niveaux supérieurs, jusqu'au niveau global.

Exemple:

```
void CalculateLWMA(int rates_total,int prev_calculated,int begin,const double &price[]
  {
   int        i,limit;
   static int weightsum=0;
   double     sum=0;
//---
   if(prev_calculated==0)
     {
      limit=MA_Period+begin;
      //--- set empty value for first limit bars
      for(i=0; i<limit; i++) LineBuffer[i]=0.0;
      //--- calculate first visible value
      double firstValue=0;
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
      firstValue/=(double)weightsum;
      LineBuffer[limit-1]=firstValue;
     }
   else
     {
      limit=prev_calculated-1;
     }

   for(i=limit;i<rates_total;i++)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      sum=0;
      for(int j=0; j<MA_Period; j++) sum+=(MA_Period-j)*price[i-j];
      LineBuffer[i]=sum/weightsum;
     }
//---
  }
```

Faites attention à la variable i, déclarée à la ligne

```
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
```

Sa portée de visibilité est seulement la boucle for, à l'extérieur de cette boucle il y a une autre variable avec le même nom, déclaré au début de la fonction. En outre, dans le corps de la boucle la variable k est declarée, dont la portée de visibilité est le corps de la boucle.

Les variables locales peuvent être déclarés avec le spécificateur d'accès static. Dans ce cas, le compilateur mit une telle variable dans le pool globale de mémoire. Pour cela la durée de vie d'une variable statique coïncide avec la durée de vie du programme. En même temps la portée de visibilité de telle variable est limitée par le bloc, dans lequel elle est déclarée.

Voir aussi

Types des données, Encapsulation et extensibilité des types, Initialisation des variables, Création et destruction des objets

© 2000-2015, MetaQuotes Software Corp.

La création et la destruction des objets

Après le chargement pour l'exécution du programme mql5 de chaque variable la mémoire est allouée en conformité du type de variable. Les variables sont divisées en deux catégories en termes d'accès - les variables globales et les variables locales, et selon les classes de mémoire: les paramètres d'entrée de programme mql5, statiques et automatiques. Si c'est nécessaire chaque variable est initialisée par la valeur correspondante. Après l'utilisation la variable déinitialise et la mémoire utilisée par elle, revient au système exécuté MQL5.

Initialisation et déinitialisation des variables globales

L'initialisation des variables globales est produit automatiquement après le chargement du mql5-programme et et jusqu'à l'appel d'aucune fonction. L'attribution des valeurs initiales aux variables des types simples est produite pendant l'initialisation et le constructeur est appelé s'il existe. Les variables d'entrée sont toujours déclarées sur le niveau global, sont initialisées par les valeurs spécifiées par les utilisateurs dans le dialogue de démarrage du programme mql5.

Malgré que des variables statiques sont déclarées d'habitude sur le niveau local, la mémoire pour ces variables est distribuée à l'avance, et l'initialisation est effectuée immédiatement après le chargement du programme, tout comme pour les variables globales.

L'ordre d'initialisation correspond à l'ordre de la déclaration de la variable dans le programme, et la deinitialisation est faite dans l'ordre inverse avant le déchargement de programme mql5. Cette règle est seulement pour les variables qui n'ont pas été créés par l'opérateur new. Ces variables sont créés et initialisés automatiquement immédiatement après le chargement et sont déinitialisées immédiatement avant la décharge du programme.

Initialisation et déinitialisation des variables locales

Si la variable déclarée au niveau local n'est pas statique, alors l'allocation de mémoire pour une telle variable se fait automatiquement. Les variables locales, ainsi que celles de globales, initialisées automatiquement lorsque le programme rencontre une déclaration de la variable locale. Ainsi, l'ordre d'initialisation correspond à l'ordre de la déclaration.

Des variables locales sont déinitialisées à la fin du bloc du programme, dans lequel ils sont déclarés et dans l'ordre inverse de leur déclaration. Le bloc du programme c'est un opérateur composé, qui puisse être la partie d'opérateur de choix switch, la boucle(for, while, do-while), le corps de fonction ou le partie d' opérateur if-else.

L'Initialisation des variables locales a lieu seulement quand l'exécution du programme atteint la déclaration de la variable. Si le bloc, dans lequel la variable est déclarée pendant l'exécution du programme, n'a pas été exécuté, donc une telle variable n'est pas initialisée.

Initialisation et déinitialisation des objets dynamiquement placés

Les indicateurs d'objets présentent le cas particulier parce que la déclaration d'indicateur n'implique pas les initialisations de l'objet existant. Les objets dynamiquement placés sont initialisés seulement au moment de la création d'exemplaire de classe par l'opérateur new. L'initialisation de l'objet envisage l'appel du constructeur de la classe correspondante. S'il n'y a pas d'aucun constructeur approprié dans la classe, donc ses membres ayant le type simple, ne seront pas initialisés automatiquement; les membres des types la chaîne, un tableau dynamique et un objet composé

© 2000-2015, MetaQuotes Software Corp.

serront initialisés automatiquement.

Les indicateurs peuvent être déclarés sur le niveau local ou global, et ils peuvent être initialisées par la valeur vide NULL par la valeur de l'indicateur de même type ou du type engendré. Si pour l'indicateur déclaré sur le niveau local, a été appelé par l'opérateur *new*, donc l'opérateur *delete* pour cet indicateur doit être exécuté avant la sortie de ce niveau. Autrement l'indicateur sera perdu et l'objet ne pourra pas être supprimé clairement.

Tous les objets créés par l'expression *l'indicateur_de_l'objet*=new *Nom_de_la_Classe*, doivent être nécessairement supprimés par la suite par l'opérateur delete(*l'indicateur_de_l'objet*). Si pour quelques raisons une telle variable à l'issue de travail de programme n'a pas été supprimé par l'opérateur delete, donc le message sera déduit au journal "Experts". Il est possible de déclarer plusieurs variables, et leur assigner l'indicateur d'un objet.

Si un objet dynamiquement créé a un constructeur, ce constructeur sera appelé au moment de l'exécution de l'opérateur *new*. Si un objet a le déstructeur, le déstructeur sera appelé au moment de l'exécution de l'opérateur *delete*.

Ainsi, les objets dynamiquement placés sont créés seulement au moment de la création par l'opérateur *new*, et ils sont supprimées garantis par l'opérateur *delete* ou par le système automatiquement exécutif MQL5 au moment de la décharge du programme. L'ordre de la déclaration des indicateurs d'objets dynamiquement crée n'influence pas à d'ordre de leur initialisation. L'ordre d'initialisation et de déinitialisation est entièrement contrôlée par le programmateur.

Les particularités du travail avec la mémoire dynamique

Au travail avec les tableaux dynamiques la mémoire libérée revient tout de suite au système.

À la création de l'objet dynamique de la classe par new, la mémoire est cherchée d'abord dans le pool de la mémoire des classes, avec qui lequel travaille le gestionnaire de la mémoire, et si dans le pool il y a pas assez de mémoire, la mémoire est demandée dans le système. À la suppression de l'objet dynamique pardelete, la mémoire occupée par l'objet, revient au pool de la mémoire des classes.

Le gestionnaire de la mémoire rend la mémoire au système à la fois après la sortie des fonctions-gestionnaires des événements: OnInit(), OnDeinit(), OnStart(), OnTick(), OnCalculate(), OnTimer(), OnTrade(), OnTester(), OnTesterInit(), OnTesterPass(), OnTesterDeinit(), OnChartEvent(), OnBookEvent().

Caractéristique brève des variables

Les renseignements principaux d'ordre de création, de destruction, d'appel de constructeur et des déstructeurs sont donnés dans le tableau bref.

	Variable automatique globale	Variable automatique locale	Objet dynamiquement crée
Initialisation	immédiatement après le chargement de programme mql5	à l'accomplissement pendant l'exécution de la ligne de code, où elle est déclarée	pendant l'exécution de l'opérateur **new**

© 2000-2015, MetaQuotes Software Corp.

Ordre d'initialisation	en ordre de la déclaration	en ordre de la déclaration	ne dépend pas de l'ordre de la déclaration
Déinitialisation	avant le décharge de programme mql5	à la sortie de l'exécution du bloc de la déclaration	pendant l'exécution de l'opérateur **delete** ou avant le démarrage du programme mql5
Ordre de déinitialisation	dans l'ordre inverse de l'initialisation	dans l'ordre inverse de l'initialisation	ne dépend pas de l'ordre de l'initialisation
Appel de constructeur	pendant le démarrage du programme mql5	pendant l'initialisation	pendant l'exécution de l'opérateur *new*
Appel de déstructeur	pendant le décharge de programme mql5	à la sortie du bloc, dans lequel la variable a été initialisée	pendant l'exécution de l'opérateur *delete*
Messages des erreurs	Message au journal "Experts" de tentative de supprimer l'objet automatiquement créé	Message au journal "Experts" de tentative de supprimer l'objet automatiquement créé	Message au journal "Experts" d'objets dynamiquement créés pas supprimés pendant le démarrage du programme mql5

Voir aussi

Types des données, Encapsulation et extensibilité des types, Initialisation des variables, Portée de visibilité et la durée de vie des variables

© 2000-2015, MetaQuotes Software Corp.

Préprocesseur

Le préprocesseur – est un sous-système spécial de compilateur MQL5, qui s'occuppe de la préparation préliminaire du texte initial du programme juste avant sa compilation.

Le préprocesseur permet améliorer la lisibilité du code initial. Le code peut être structuré par l'inclusion de fichiers spécifiques avec les codes initiales de programmes mql5. La possibilité d'allouer les noms mnémotechniques à constantes spécifiques contribue à l'amélioration de la lisibilité de code.

Le préprocesseur permet de définir des paramètres spécifiques de programmes mql5:

- Déclarer les constantes
- Installer les propriétés du programme
- Inclure des fichiers au texte du programme
- Importer les fonctions
- Использовать условную компиляцию

Si on utilise le caractère # comme le premier caractère dans la chaîne de programme donc cette chaîne est une directive de préprocesseur. La directive de préprocesseur se termine par le caractère de transfert à la nouvelle ligne.

© 2000-2015, MetaQuotes Software Corp.

La déclaration de constante (#define, #undef)

La directive #define peut être utilisé pour assigner des noms mnémoniques aux constantes. Il y a deux formes:

```
#define identifier expression                    // la forme sans paramètre
#define identifier(par1,... par8) expression     // la forme paramétrique
```

La directive #define substitue **expression** au lieu des inclusions trouvées séquentielles identifier dans le texte initial. identifier est remplacé seulement dans le cas s'il présente le token séparé. identifier n'est pas remplacée si elle est la partie du commentaire, la partie de la chaîne ou la partie d'autre identificateur le plus longue.

L'indificateur de la constante est soumis aux mêmes règles, que pour les noms de variables. La valeur peut être n'importe quel type:

```
#define ABC               100
#define PI                3.14
#define COMPANY_NAME      "MetaQuotes Software Corp."
...
void ShowCopyright()
  {
   Print("Copyright  2001-2009, ",COMPANY_NAME);
   Print("http://www.metaquotes.net");
  }
```

expression peut être composé de plusieurs token, comme les mots-clés, des constantes, les expression constantes et non-constantes. **expression** se termine avec la fin de la ligne et ne peut être reporté à la ligne suivante.

Exemple:

```
#define TWO         2
#define THREE       3
#define INCOMPLETE TWO+THREE
#define COMPLETE   (TWO+THREE)
void OnStart()
  {
   Print("2 + 3*2 = ",INCOMPLETE*2);
   Print("(2 + 3)*2 = ",COMPLETE*2);
  }
/* Résultat
   2 + 3*2 = 8
   (2 + 3)*2 = 10
*/
```

La forme paramétrique #define

A la forme paramétrique toutes les occurrences suivantes trouvées "identifier" seront remplacés sur "expression" en tenant compte des paramètres réels. Par exemple,

© 2000-2015, MetaQuotes Software Corp.

```
 // l'exemple avec deux paramètres a et b
#define A 2+3
#define B 5-1
#define MUL(a, b) ((a)*(b))

double c=MUL(A,B);
Print("c=",c);
/*
l'expression double c=MUL(A,B);
est égal au double c=((2+3)*(5-1));
*/
// Le résultat
// c=20
```

Concluez obligatoirement les paramètres aux parenthèses à l'utilisation des paramètres dans "expression", puisque cela permet d'éviter les erreurs non évidentes, qu'il est difficile de trouver. Si on écrit l'exemple sans utilisation des parenthèses, le résultat sera tout à fait autre:

```
 // l'exemple avec deux paramètres a et b
#define A 2+3
#define B 5-1
#define MUL(a, b) a*b

double c=MUL(A,B);
Print("c=",c);
/*
l'expression double c=MUL(A,B);
est égal au double c=2+3*5-1;
*/
// Le résultat
// c=16
```

Pas plus de 8 paramètres sont admis à l'utilisation de la forme paramétrique.

```
 // une forme paramétrique correcte
#define LOG(text)  Print(__FILE__,"(",__LINE__,") :",text)   // un paramètre - 'text'

 //une forme paramétrique incorrecte
#define WRONG_DEF(p1, p2, p3, p4, p5, p6, p7, p8, p9)   p1+p2+p3+p4 // plus de 8 paran
```

Директива #undef

Директива #undef предназначена для отмены макроса, объявленного ранее.

Exemple:

```
#define MACRO

void func1()
  {
#ifdef MACRO
   Print("MACRO is defined in ",__FUNCTION__);
#else
   Print("MACRO is not defined in ",__FUNCTION__);
```

© 2000-2015, MetaQuotes Software Corp.

```
#endif
  }

#undef MACRO

void func2()
  {
#ifdef MACRO
   Print("MACRO is defined in ",__FUNCTION__);
#else
   Print("MACRO is not defined in ",__FUNCTION__);
#endif
  }

void OnStart()
  {
   func1();
   func2();
  }

/* Le résultat:
 MACRO is defined in func1
 MACRO is not defined in func2
*/
```

Voir aussi

Identificateurs, Constantes de caractères

© 2000-2015, MetaQuotes Software Corp.

Les propriétés des programmes (#property)

On peut spécifier les paramètres spécifiques supplémentaires *#property*, dans chaque programme mql5 qui aident au terminal de client de servir correctement les programmes sans la nécessité de les démarrer. Tout d'abord cela concerne une configuration extérieure des indicateurs. Les propriétés décrites dans le fichier inclus, sont totalement ignorés. Les propriétés doivent être spécifiées dans le fichier mq5 principal.

```
#property identificateur  valeur
```

Le compilateur écrira des valeurs déclarées dans la configuration du module exécuté.

Constante	Type	Description
icon	string	Le chemin vers le fichier avec une image qui sera affichée pour le programme EX5, les règles du chemin sont les mêmes que pour les ressources. La propriété doit être indiquée dans un module principal avec le code initial MQL5. Le fichier de l'icône doit être dans le format ICO.
link	string	La lien au site de la compagnie-producteur
copyright	string	Le nom de la compagnie-producteur
version	string	La version du programme, pas plus 31 caractères
description	string	La description brève de texte du programme mql5. Plusieurs descriptions peuvent être présentes, chacune décrrit une ligne du texte. La longueur totale de toutes description ne peut pas dépasser 511 en tenant compte de transfert des lignes
stacksize	int	Указывает размер стека для MQL5 программы, стек достаточного объема требуется в случае выполнения рекурсивных вызовов функций. При запуске скрипта или эксперта на графике

© 2000-2015, MetaQuotes Software Corp.

		выделяется стек не менее 8Мб, для индикаторов свойство не работает - стек всегда фиксированного объема в 1Мб. При запуске в тестере программе всегда выделяется стек в размере 16 Мб.
library		La bibliothèque; aucune fonction de début n'est pas assignée, On peut importer les fonctions avec le modificateur export dans les autres programme mql5
indicator_applied_price	int	Spécifie la valeur par défaut pour le champ "Apply to". On peut spécifier l'une des valeurs d'énumération ENUM_APPLIED_PRICE. Si la propriété n'est pas spécifiée, la valeur s'applique par défaut PRICE_CLOSE
indicator_chart_window		Visualiser l'indicateur à la fenêtre du graphique
indicator_separate_window		Visualiser l'indicateur à la fenêtre séparée
indicator_height	int	La hauteur fixée de sous-fenêtre de l'indicateur en pixels (la propriété INDICATOR_HEIGHT)
indicator_buffers	int	Le nombre de tampons pour le calcul de l'indicateur
indicator_plots	int	Le nombre de séries graphiques dans l'indicateur
indicator_minimum	double	Le bornage inférieur de l'échelle de la fenêtre séparée de l'indicateur
indicator_maximum	double	Le bornage supérieur de l'échelle de la fenêtre séparée de l'indicateur
indicator_labelN	string	Met une étiquette pour la N-ième série graphique, affichée dans la fenêtre DataWindow. Pour la série graphique

© 2000-2015, MetaQuotes Software Corp.

		exigeant quelques tampons d'indicateur (DRAW_CANDLES, DRAW_FILLING et les autres), les noms d'étiquettes sont spécifiées par le séparateur ';'.
indicator_colorN	color	La couleur pour la sortie de la ligne N, où N - est le numéro de la série graphique; numération de 1
indicator_widthN	int	Lépaisseur de la ligne dans la série graphique, où N - est le numéro de la série graphique; numération de 1
indicator_styleN	int	Le style de la ligne dans la série graphique, spécifié à l'aide de la valeur de ENUM_LINE_STYLE. N - le numéro de la série graphique, numération de 1
indicator_typeN	int	Construction graphique est spécifiée par la valeur de ENUM_DRAW_TYPE. où N - est le numéro de la série graphique; numération de 1
indicator_levelN	double	Un niveau horizontale de N dans une fenêtre séparée d'indicateur
indicator_levelcolor	color	La couleur des niveaux horizontaux de l'indicateur
indicator_levelwidth	int	L'épaisseur des niveaux horizontaux de l'indicateur
indicator_levelstyle	int	Le style des niveaux horizontaux de l'indicateur
script_show_confirm		Visualiser la fenêtre de la confirmation avant le lancement du script
script_show_inputs		Visualiser la fenêtre avec les propriétés avant le lancement du script et interdire la sortie de la fenêtre de la confirmation
tester_indicator	string	Le nom de l'indicateur d'utilisateur dans le format "*le nom_de l'indicateur.ex5*". Les

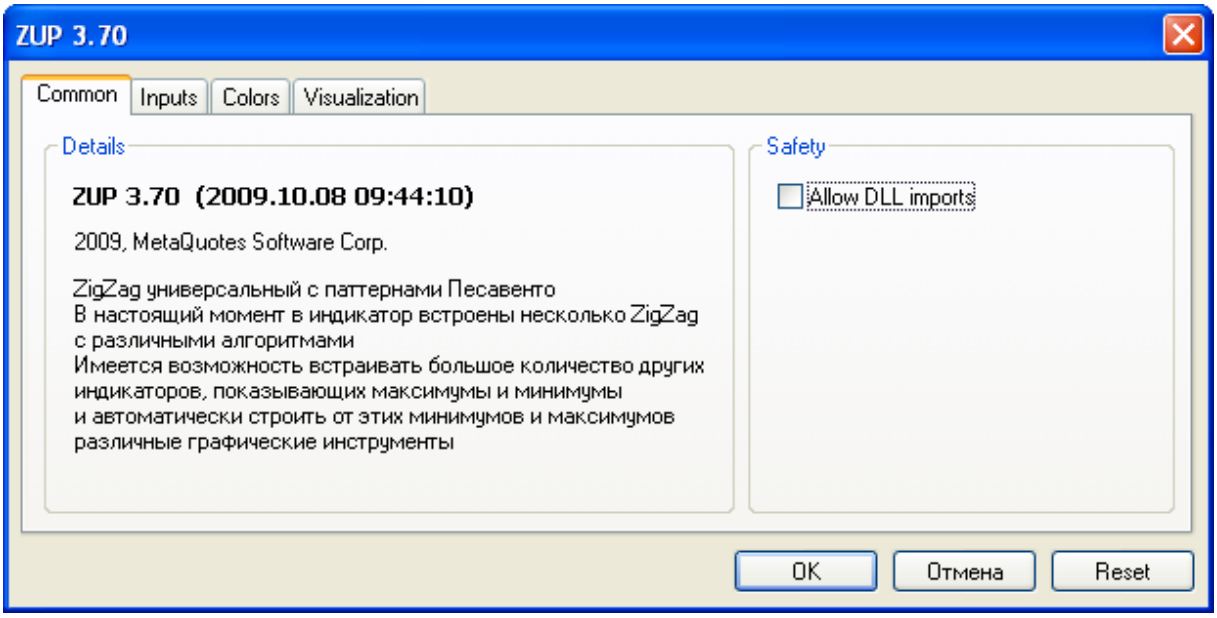
© 2000-2015, MetaQuotes Software Corp.

		indicateurs nécessaires au test sont définis automatiquement de l'appel des fonctions, iCustom(), si le paramètre correspondant est spécifié par la ligne constante. Pour les autres cas (l'utilisation de la fonction IndicatorCreate() ou l'utilisation de la ligne non constante dans le paramètre qui spécifie le nom de l'indicateur) la propriété donnée est nécessaire
tester_file	string	Le nom du fichier pour le testeur avec l'indication de l'extension, conclu aux guillemets doubles (comme la ligne constante). Le fichier indiqué sera transmis au testeur au travail. Les fichiers d'entrée pour le test, s'ils sont nécessaires, doivent être indiqués toujours
tester_library	string	Le nom de la bibliothèque avec l'extension, conclu aux guillemets doubles. La bibliothèque peut être avec l'extension dll, ainsi qu'avec l'extension ex5. Les bibliothèques nécessaires au test sont définies automatiquement. Cependant, si quelque bibliothèque est utilisée par l'indicateurd'utilisateuril est nécessaire d'utiliser la propriété donnée

Exemple de la tâche de description et le numéro de version

```
#property version     "3.70"      // version actuelle de l'expert
#property description "ZigZag universel avec les patterns Pesavento"
#property description "A présent dans l'indicateur sont insérés quelques ZigZag avec l
#property description "Il y a la possibilité d'intégrer un grand nombre d'autres indic
#property description "minimum et automatiquement construire de ces minimums et les ma
```

© 2000-2015, MetaQuotes Software Corp.



L'exemple de l'indication de la balise séparée pour chaque tampon d'indicateur ("C open;C high;C low;C close")

```
#property indicator_chart_window
#property indicator_buffers 4
#property indicator_plots   1
#property indicator_type1   DRAW_CANDLES
#property indicator_width1  3
#property indicator_label1  "C open;C high;C low;C close"
```

© 2000-2015, MetaQuotes Software Corp.

L'inclusion de fichiers (#include)

La ligne de commande *#include* peut se rencontrer en n'importe quelle place du programme, mais d'habitude toutes les inclusions sont placées dans le début du fichier du texte initial. Le format de l'appel:

```
#include <nom_de fichier>
#include "nom_de fichier"
```

Exemples:

```
#include <WinUser32.mqh>
#include "mylib.mqh"
```

Le préprocesseur remplace la ligne *#include <nom_de fichier>* par le contenu du fichier WinUser32.mqh. Les chevrons indiquent que le fichier WinUser32.mqh sera pris du répertoire standard (D'habitude c'est *le répertoire_de terminal\MQL5\Include*). Le répertoire actuel n'est pas examiné.

Si le nom du fichier est entre les guillemets, la recherche s'effectue dans le répertoire courant (qui contient le fichier principal du texte initiale). Le répertoire standard n'est pas examiné.

Voir aussi

Standard Library, Importation des fonctions

© 2000-2015, MetaQuotes Software Corp.

L'importation des fonctions (#import)

L'importation des fonctions se réalise de modules MQL5 compilés (les fichiers *.ex5) et des modules de système d'exploitation (les fichiers *.dll). Le nom du module est spécifié dans la directive *#import*. Pour que le compilateur puisse correctement régulariser l'appel de la fonction importée et organiser une correcte transmission des paramètres, la description complète des fonctions est nécessaire. Les descriptions de fonctions suivent immédiatement la directive *#import "nom du module"*. Une nouvelle commande *#import* (peut être sans paramètres) termine le bloc de la description des fonctions importées.

```
#import "nom_du fichier"
    func1 define;
    func2 define;
    ...
    funcN define;
#import
```

Les fonctions importées peuvent avoir n'importe quels noms. Il est possible d'importer simultanément les fonctions avec des noms identiques des modules différents. Les fonctions importées peuvent avoir les noms coïncidant avec les noms des fonctions insérées. L'opération de la résolution du contexte définit, quelle fonction doit être appelée.

L'ordre de la recherche du fichier indiqué après le mot-clé#import, est décrit dans le paragraphe Appel des fonctions importées.

Puisque les fonctions importées se trouvent en dehors de module compilé, le compilateur ne peut pas vérifier la validité des paramètres transmis. C'est pourquoi, pour éviter les erreurs, il est nécessaire décrire avec précision la composition et l'ordre des paramètres transmis aux fonctions importées. Les paramètres transmis aux fonctions importées (de EX5, ainsi que des modules DLL), peuvent avoir des valeurs par défaut.

Dans les fonctions importées à titre des paramètres on ne peut pas utiliser:

- les indicateurs (*);
- les liens aux objets, contenant les tableaux dynamiques et/ou les indicateurs.

Aux fonctions importées de DLL on ne peut pas transmettre à titre du paramètre les classes, le tableau des lignes ou les objets complexes contenant les lignes et/ou les tableaux dynamiques de n'importe quels types.

Exemples:

```
#import "user32.dll"
int    MessageBoxW(uint hWnd,string lpText,string lpCaption,uint uType);
#import "stdlib.ex5"
string ErrorDescription(int error_code);
int    RGB(int red_value,int green_value,int blue_value);
bool   CompareDoubles(double number1,double number2);
string DoubleToStrMorePrecision(double number,int precision);
string IntegerToHexString(int integer_number);
#import "ExpertSample.dll"
int    GetIntValue(int);
```

© 2000-2015, MetaQuotes Software Corp.

```
double GetDoubleValue(double);
string GetStringValue(string);
double GetArrayItemValue(double &arr[],int,int);
bool   SetArrayItemValue(double &arr[],int,int,double);
double GetRatesItemValue(double &rates[][6],int,int,int);
#import
```

Pour l'importation des fonctions pendant l'exécution du programme mql5 on utilise un binding précoce. Cela signifie que la bibliothèque est chargée en train du chargement du programme ex5 qui l'utilise.

Il n'est pas recommandé d'utiliser le nom entièrement qualifié du module chargeable du type *Drive:\Directory\FileName.Ext*. Les bibliothèques MQL5 sont chargées du dossier *terminal_dir\MQL5\Libraries*.

Voir aussi

Inclusion de fichiers

© 2000-2015, MetaQuotes Software Corp.

Условная компиляция (#ifdef, #ifndef, #else, #endif)

Директивы условной компиляции препроцессора позволяют компилировать или пропускать часть программы в зависимости от выполнения некоторого условия.

Условие может принимать одну из описываемых ниже форм.

```
#ifdef identifier
   // код, находящийся здесь, компилируется, если identifier уже был определен для пре
#endif
```

```
#ifndef identifier
   // код, находящийся здесь, компилируется, если identifier в данный момент не опреде
#endif
```

За любой из команд условной компиляции может следовать произвольное число строк, содержащих, возможно, команду вида #else и заканчивающихся #endif. Если проверяемое условие справедливо, то строки между #else и #endif игнорируются. Если же проверяемое условие не выполняется, то игнорируются все строки между проверкой и командой #else, а если ее нет, то командой #endif.

Пример:

```
#ifndef TestMode
   #define TestMode
#endif
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef TestMode
      Print("Test mode");
   #else
      Print("Normal mode");
   #endif
  }
```

В зависимости от типа программы и режима компиляции стандартные макросы определяются следующим образом:

Макрос __MQL5__ доступен при компиляции файла *.mq5, при компиляции *.mq4 доступен макрос __MQL4__.
Макрос _DEBUG доступен при компиляции под отладку.
Макрос _RELEASE доступен при компиляции не под отладку.

Пример:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2015, MetaQuotes Software Corp.

```
{
 #ifdef __MQL5__
    #ifdef _DEBUG
       Print("Hello from MQL5 compiler [DEBUG]");
    #else
      #ifdef _RELEASE
         Print("Hello from MQL5 compiler [RELEASE]");
      #endif
   #endif
 #else
    #ifdef __MQL4__
       #ifdef _DEBUG
         Print("Hello from MQL4 compiler [DEBUG]");
      #else
         #ifdef _RELEASE
            Print("Hello from MQL4 compiler [RELEASE]");
         #endif
      #endif
   #endif
 #endif
}
```

© 2000-2015, MetaQuotes Software Corp.

La programmation orientée objet

La programmation orientée objet (POO)- cette programmation s'est concentrée sur des données, les données et le comportement sont inséparablement liés. Les données et les comportements constituent une classe, mais les objets sont les exemplaires de la classe.

Les composantes de l'approche orientée objet sont:

- Encapsulation et extensibilité des types
- Héritage
- Polymorphisme
- Surcharge
- Fonctions virtuelles

La programmation orientée objet considère le calcul comme la modélisation de comportement. L'article modelé est l'objet représenté par des abstractions informatiques. Supposons que nous voulons écrire un jeu bien connu "Tetris", pour cela nous devons apprendre à modeler l'apparition de figure accidentelle faite de quatre cubes joint l'un avec l'autre par les côtes. Il faut régler aussi la vitesse de la chute de la figure, définir l'opération de la rotation et le décalage de la figure. Les navigations de la figure sur l'écran sont limitées par les marges du verre, nous devons modeler cette exigence aussi. En outre des séries remplies de cubes dans le verre doivent être supprimées et il faut mener le compte des points gagnés dans le jeu.

Ainsi, un tel jeu simple dans la compréhension demande la création de quelques modèles - les modèles de la figure, le modèle du verre, le modèle du mouvement de la figure dans le verre etc. Tous ces modèles sont les abstractions présentées par les calculs dans l'ordinateur. Pour la description de tels modèles appliquent la notion *le type abstrait de données*, TAD (ou le type composé des données). Strictement dit, le modèle du mouvement de "la figure" dans "le verre" n'est pas le type de données, mais est l'ensemble des opérations sur les données comme "la figure", utilisant les limitations des données du type "le verre".

Les objets sont les variables de la classe. La programmation orientée objet permet facilement de créer et utiliser TAD. La programmation orientée objet utilise le mécanisme de la héritage. La héritage est avantageuse pour ce que permet de recevoir les types dérivés des types de données déjà définis par l'utilisateur. Ainsi, pour la création des figures dans le tetris est confortable d'abord de créer la classe de base Shape, à la base de laquelle sont reçues les types variables toutes les sept figures possibles dans le tetris. Dans la classe de base on définit la conduite des figures, mais dans les dérivées on précise l'implémentation du comportement de chaque figure concrète.

Dans la POO les objets sont responsables de leur comportement. Le programmeur de TAD devrait inclure un code pour décrire n'importe quel comportement qui serait normalement attendu des objets correspondants. Ce que l'objet lui-même répond pour son comportement, simplifie considérablement la tâche de la programmation pour l'utilisateur de cet objet.

Si nous voulons dessiner la figure sur l'écran, nous devons connaître où il y aura son centre et comment la dessiner. Si la figure séparée comprend parfaitement, comme se dessiner, le programmeur à l'utilisation d'une telle figure doit seulement transmettre à l'objet le message "être dessiné".

Le langage MQL5 ressemble à C++, et il a aussi le mécanisme d'encapsulation pour la réalisation de TAD. L'encapsulation reunit dans lui-même, d'une part, les détails intérieurs de l'implémentation du

© 2000-2015, MetaQuotes Software Corp.

type concret et, avec l'autre, les fonctions accessibles du dehors, qui peuvent agir sur les objets de ce type. Les détails de l'implémentation peuvent être inaccessible pour le programme, qui utilise ce type.

A la notion de POO se rapporte l'ensemble entier des conceptions, y compris les suivants:

- La simulation des actions du monde réel
- La présence des types de données définis par l'utilisateur
- La dissimulation des détails de l'implémentation
- La possibilité de l'utilisation multiple du code grâce à la héritage
- L'interprétation des appels des fonctions à l 'étape de l'exécution

Certains de ces concepts sont assez vagues, certains sont abstraits, les autres sont du caractère général.

© 2000-2015, MetaQuotes Software Corp.

L'encapsulation et l'extensibilité des types

La POO – est une approche équilibrée à l'écriture de logiciels. Les données et le comportement sont emballés ensemble. Cette encapsulation crée des types de données définis par l'utilisateur en étendant les types de données du langage et en communiquant avec eux. L'extensibilité des types – c'est la possibilité d'ajouter au langage les types des données définis par l'utilisateur, qui sont aussi faciles à utiliser, ainsi que les types de base.

Le type de données abstrait, par exemple, la ligne, est la description idéal du comportement du type connu à tout le monde. L'utilisateur de la ligne sait que les opérations, telles que la concaténation ou l'impression ont un certain comportement. Les opérations de la concaténation ou de l'impression s'appellent les méthodes.

L'implémentation concrète de TAD peut avoir quelques limitations; par exemple, les lignes peuvent être limitées sur la longueur. Ces limitations affectent au comportement ouvert à tous. En même temps, les détails intérieurs ou fermés de l'implémentation n'influencent pas directement, à la façon comment l'utilisateur voit l'objet. Par exemple, la ligne se réalise souvent comme le tableau; pendant que l'adresse de base intérieure des éléments de ce tableau et son nom ne sont pas essentielle pour l'utilisateur.

L'encapsulation est la capacité à cacher les détails intérieurs lorsque l'interface ouverte vers le type défini par l'utilisateur est fourni. Dans MQL5, aussi bien que dans C++, des définitions de classe et de structure (class et struct) sont utilisés dans la combinaison avec les mots d'accès private (fermé), protected (protégé) et public (ouvert) pour assurer l'encapsulation.

Le mot clé public montre que l'accès aux membres, qui se trouvent après celui-ci, est ouvert sans aucune limitation. Sans ce mot clé les membres de la classe sont fermés par défaut. Les membres fermés sont accessibles seulement aux fonctions-les membres seulement de sa classe.

Les membres protégés de la classe sont accessibles aux fonctions- les membres non seulement de sa classe, mais aussi des classes-héritières. Les membres ouverts sont accessibles à n'importe quelle fonction à l'intérieur du domaine de la visibilité de la déclaration de la classe. La protection permet de cacher la partie de l'implémentation de la classe, en prévenant alors les changements imprévus de la structure des données. La limitation d'accès ou la dissimulation des données est la particularité de la programmation orientée objet.

D'habitude on tâche de protéger les membres de la classe et les déclarer avec le modificateur protected, l'installation et la lecture des valeurs de ces membres se réalise avec l'aide des méthodes spéciales set- et get, qui sont définies avec le modificateur de l'accès public.

Exemple:

```
class CPerson
  {
protected:
   string            m_name;                    // nom
public:
   void              SetName(string n){m_name=n;}// établit le nom
   string            GetName(){return (m_name);} // rend le nom
  };
```

Une telle approche donne quelques avantages. Premièrement, par le nom de fonction on peut

© 2000-2015, MetaQuotes Software Corp.

comprendre ce qu'elle fait - établit ou reçoit la valeur d'un membre de classe. Deuxièmement, probablement dans le futur il faudra changer le type de la variable m_name dans la classe CPerson ou dans quelque des ces dérivées des classes.

Dans ce cas-là, il faut juste changer l'implémentation de fonctions SetName() et GetName(), pendant que les objets de la classe de CPerson seront disponibles pour utiliser dans un programme sans changements dans le code, parce que l'utilisateur même ne saura pas que le type de données m_name a changé.

Exemple:

```
struct Name
  {
   string            first_name;                  // prénom
   string            last_name;                   // nom
  };

class CPerson
  {
protected:
   Name              m_name;                      // prénom
public:
   void              SetName(string n);
   string            GetName(){return(m_name.first_name+" "+m_name.last_name);}
private:
   string            GetFirstName(string full_name);
   string            GetLastName(string full_name);
  };

void CPerson::SetName(string n)
  {
   m_name.first_name=GetFirstName(n);
   m_name.last_name=GetLastName(n);
  }

string CPerson::GetFirstName(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>0) StringSetCharacter(full_name,pos,0);
   return(full_name);
  }

string CPerson::GetLastName(string full_name)
  {
   string ret_string;
   int pos=StringFind(full_name," ");
   if(pos>0) ret_string=StringSubstr(full_name,pos+1);
   else      ret_string=full_name;
   return(ret_string);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Voir aussi

Types des données

© 2000-2015, MetaQuotes Software Corp.

L'héritage

La particularité de la POO est l'encouragement de réutilisation codée au cours de l'héritage. Une nouvelle classe est produite d'existante, appelée comme la classe de base. La classe dérivée utilise les membres de la classe de base, mais peut aussi les changer et compléter.

Les plusieurs types représentent les versions aux sujets existants. Il est souvent fatiguant de développer un nouveau code pour chacun d'entre eux. En plus, un nouveau code implique de nouvelles erreurs. La classe dérivée hérite de la description de la classe de base, en faisant inutile l'élaboration réitérée et le test du code. Les rapports d'héritage sont hiérarchiques.

La hiérarchie est la méthode, permettant de copier les éléments dans toute leur diversité et la complexité. Elle introduit la classification des objets. Par exemple, dans la classification périodique des éléments chimiques ce sont des gaz. Ils possèdent les propriétés inhérentes à tous les éléments du système.

Les gaz inertes est une importante sous-classe suivante. La hiérarchie consiste en ce que le gaz inerte, par exemple, l'argon est un gaz, mais le gaz, est à son tour l'élément du système. Une telle hiérarchie permet facilement d'interpréter le comportement des gaz inertes. Nous connaissons que leurs atomes contiennent les protons et les électrons ce qu'est exacte et pour les autres éléments.

Nous connaissons qu'ils demeurent dans l'état gazeux à la température ordinaire, comme tous les gaz. Nous connaissons qu'aucun gaz de la sous-classe des gaz inertes n'entre pas dans la réaction ordinaire chimique avec un des éléments, et cette propriété est celle de tous les gaz inertes.

Examinons l'héritage à l'exemple des figures géométriques. Pour la description de toute la diversité des figures simples (le cercle, le triangle, le rectangle, le carré etc) c'est mieux de créer la classe de base (TAD), qui est le prédécesseur de toutes les classes dérivées.

Créons la classe de base CShape,dans laquelle il y a seulement des termes généraux décrivant la figure. Ces membres décrivent les propriétés inhérentes à n'importe quelle figure - le type de la figure et la coordonnée du point principal du rattachement.

Exemple:

```
//--- Classe de base Figure
class CShape
  {
protected:
   int       m_type;                    // type de figure
   int       m_xpos;                    // X - coordonnée du point du rattachement
   int       m_ypos;                    // Y - coordonnée du point du rattachement
public:
             CShape(){m_type=0; m_xpos=0; m_ypos=0;} // constructeur
   void      SetXPos(int x){m_xpos=x;} // établissons X
   void      SetYPos(int y){m_ypos=y;} // établissons Y
  };
```

Ensuite créons de la classe de base les classes dérivées, auxquelles ajoutons les champs nécessaires précisant chaque classe concrète. Pour la figure Circle(le cercle) il est nécessaire d'ajouter le membre, qui contient la valeur de rayon. La figure Quadrate (le carré) se caractérise par la valeur de la côté du carré. C'est pourquoi les classes dérivées, héritées de la classe de base CShape seront déclarées ainsi:

© 2000-2015, MetaQuotes Software Corp.

```
//--- la classe dérivée Le cercle
class CCircle : public CShape        // Après les deux-points on indique la classe de
  {                                    // de quelle on produit la héritage
private:
   int             m_radius;           // rayon du cercle

public:
                   CCircle(){m_type=1;}// constructeur, le type est égual à 1
  };
```

Pour la forme Carrée la déclaration de classe est semblable:

```
//--- la classe dérivée Le carré
class CSquare : public CShape        // après les deux-points on indique la classe de
  {                                    // de quelle on produit la héritage
private:
   int            m_square_side;       // côté de carré

public:
                   CSquare(){m_type=2;} // constructeur, le type est égual à 2
  };
```

Il est nécessaire de noter que pendant la création d'objet tout d'abord on appelle le constructeur de la classe de base, après le constructeur de la classe derivée. Quand l' objet est supprimé d'abord on appelle le déstructeur de la classe derivée, et après le déstructeur de la classe de base.

Ainsi, ayant déclaré dans la classe de base les termes plus généraux, dans les classes dérivées nous pouvons ajouter les termes supplémentaires, qui précisent la classe concrète. L'héritage permet de créer de puissantes bibliothèques du code, que l'on peut utiliser plusieurs fois et de nouveau.

La syntaxe de création de classe dérivée de la classe déjà existante apparaît comme ça:

```
class nom_de la classe :
          (public | protected | private) opt  nom_de la classe_de base
  {
    déclaration des membres
  };
```

Un des aspects de la classe dérivée est la visibilité (ouverture) de ses membres-successeurs. Les mots clé public, protected et private sont utilisés pour cette indication, autant les membres de la classe de base seront accessibles pour le dérivé. L'utilisation dans le titre de la classe dérivée de la classe clé public, suivant les deux-points, signifie que les membres protégés et ouverts (protected et public) de la classe de base CShape doivent être hérités comme les membres protégés et ouverts de la classe dérivée CCircle.

Les membres fermés de la classe de base sont inaccessibles à la classe dérivée. L'héritage ouverte signifie aussi que les classes dérivées (CCircle et CSquare) sont CShape. C'est-à-dire, le carré (CSquare) est la figure (CShape), mais la figure ne doit pas être absolument le carré.

La classe dérivée est la modification de la classe de base; il hérite les membres protégés et ouverts de la classe de base. Seulement les constructeurs et les déstructeurs ne peuvent pas s'hériter. Souvent à la classe dérivée on ajoute les nouveaux membres en complément des membres de la classe de base.

© 2000-2015, MetaQuotes Software Corp.

La classe dérivée peut couper l'implémentation des fonctions-membres differente de la classe de base. Cela n'a pas rien de commun avec la surcharge, quand le sens du même nom de la fonction peut être divers pour de différentes signatures.

Dans l'héritage protégé, les membres publics et protégés de classe basée deviennent des membres protégés de classe dérivée. Dans l'héritage privé, les membres publics et protégés de classe basée deviennent des membres privés de la classe dérivée.

Aux héritages protégés et fermés ce n'est pas la vraie que l'objet de la classe dérivée est l'objet de la classe de base. L'héritage protégé et privé sont rares et chacun d'entre eux a besoin d'être utilisé soigneusement.

Il faut comprendre que le type de l'héritage (public, protected ou private) n'influence aucunement les moyens **de l'accès aux membres des classes de base dans la hiérarchie de l'héritage du descendant (le successeur)**. A n'importe quel type de l'héritage des classes dérivées seront accessibles seulement les membres de la classe de base, déclarés avec les spécificateurs de l'accès public et protected. Montrons cela à l'exemple:

```
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| La classe-exemple avec plusieurs types d'accès                   |
//+------------------------------------------------------------------+
class CBaseClass
  {
private:             //--- un membre fermé est inaccessible des descendants
   int               m_member;
protected:           //--- la méthode protégée est accessible de la classe de base et
   int               Member(){return(m_member);}
public:              // le constructeur de la classe est accessible à tous
                     CBaseClass(){m_member=5;return;};
private:             // une méthode fermée pour attribuer la valeur au membre m_member
   void              Member(int value) { m_member=value;};

  };
//+------------------------------------------------------------------+
//| La classe dérivée avec les erreurs                               |
//+------------------------------------------------------------------+
class CDerivaed: public CBaseClass // On peut ne pas indiquer l'héritage public, il va
  {
public:
   void Func() // définissons dans le descendant la fonction avec les appels aux membr
     {
      //--- la tentative de la modification du membre fermé de la classe de base
      m_member=0;         // l'erreur, le membre fermé de la classe de base n'est pas a
      Member(0);          // l'erreur, la méthode fermée de la classe de base n'est pas
      //---la lecture du membre de la classe de base
      Print(m_member);    // l'erreur, le membre fermé de la classe de base n'est pas a
      Print(Member());    // il n'y a pas d'erreur, une méthode ouverte de la classe de
```

© 2000-2015, MetaQuotes Software Corp.

```
    }
  };
```

Dans cet exemple la classe CBaseClass a seulement une méthode publique - c'est le constructeur. Les constructeurs sont appelés automatiquement à la création de l'objet de la classe. C'est pourquoi on ne peut pas s'adresser du dehors ni au membre fermé m_member, ni vers une méthode protégée Member(). Mais en cela à l'héritage ouvert (public) la mèthode Member() de la classe de base sera accessible des descendants.

A l'héritage protégé (protected) tous les membres de la classe de base avec l'accès ouvert et protégé deviennent protégés. Cela signifie que si les membres-données ouverts et les méthodes de la classe de base étaient accessibles du dehors, à l'héritage protégé ils sont accessibles maintenant seulement des classes du descendant et ses classes ultérieures dérivées.

```
//+------------------------------------------------------------------+
//| La classe-exemple avec plusieurs types d'accès                   |
//+------------------------------------------------------------------+
class CBaseMathClass
  {
private:             //--- un membre fermé est inaccessible des descendants
   double            m_Pi;
public:              //--- la réception et l'installation de la valeur pour m_Pi
   void              SetPI(double v){m_Pi=v;return;};
   double            GetPI(){return m_Pi;};
public:              // le constructeur de la classe est accessible à tous
                     CBaseMathClass() {SetPI(3.14);  PrintFormat("%s",__FUNCTION__);};
  };
//+------------------------------------------------------------------+
//| La classe dérivée, où on ne peut pas déjà changer m_Pi           |
//+------------------------------------------------------------------+
class CProtectedChildClass: protected CBaseMathClass // l'héritage protégé
  {
private:
   double            m_radius;
public:              //--- les méthodes ouvertes dans le descendant
   void              SetRadius(double r){m_radius=r; return;};
   double            GetCircleLength(){return GetPI()*m_radius;};
  };
//+------------------------------------------------------------------+
//| La fonction du lancement du script                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- à la création du descendant le constructeur de la classe de base est appelé auto
   CProtectedChildClass pt;
//--- spécifions le rayon
   pt.SetRadius(10);
   PrintFormat("Length=%G",pt.GetCircleLength());
//--- si on va commenter la ligne qui est plus bas, recevons l'erreur à l'étape de la
// pt.SetPI(3);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- et maintenant nous déclarons la variable de la classe de base et essayons de spé
   CBaseMathClass bc;
   bc.SetPI(10);
//--- voir le résultat
   PrintFormat("bc.GetPI()=%G",bc.GetPI());
  }
```

Dans cet exemple est montré que les méthodes SetPI () et GetPi () dans la classe de base CBaseMathClass sont ouvertes et sont accessibles pour l'appel de n'importe quelle place du programme. Mais en même temps pour son descendant CProtectedChildClass on peut faire les appels de ces méthodes seulement des méthodes de la classe CProtectedChildClass ou ses descendants.

A l'héritage fermé (private) tous les membres de la classe de base avec l'accès public et protected deviennent fermés et à l'héritage suivant l'appel à eux devient impossible.

Dans MQL5 il n'y a pas aucun héritage multiple.

Voir aussi

Srtuctures et Classes

© 2000-2015, MetaQuotes Software Corp.

Le polymorphisme

Le polymorphisme est une possibilité pour les objets de différentes classes liées à l'aide de l'héritage, réagir par un divers moyen à l'appel à la même fonction-élément. Cela aide à créer les mécanismes universels décrivant le comportement non seulement la classe de base, mais aussi les classes-descendantes.

Continuons à développer une classe de base CShape, dans laquelle définissons la fonction-membre GetArea(), destinée pour le calcul de la surface de la figure. Dans toutes les classes-descendantes, produites par l'héritage de la classe de base, nous redéfinissons cette fonction conformément aux règles du calcul de la surface de la figure.

Pour le carré (la classe CSquare) la surface est calculé par les côtés, pour le cercle (la classe CCircle) la surface est calculé par le rayon etc. Nous pouvons créer le tableau pour le stockage des objets comme CShape, dans lequel on peut conserver l'objet de la classe de base et ses descendants. Après nous pouvons appeler la même fonction pour chaque élément de ce tableau.

Exemple:

```
//--- Classe de base
class CShape
  {
protected:
   int            m_type;                 // type de figure
   int            m_xpos;                 // X - coordonnée du point du rattachement
   int            m_ypos;                 // Y - coordonnée du point du rattachement
public:
   void           CShape(){m_type=0;};    // constructeur, le type est égal à 0
   int            GetType(){return(m_type);};// rend le type de figure
virtual
   double         GetArea(){return (0); }// rend la surface de figure
  };
```

Maintenant toutes les classes dérivées ont la fonction-membre getArea(), qui rend une valeur zéro. L'implémentation de cette fonction dans chaque descendant se variera.

```
//--- classe dérivée Circle
class CCircle : public CShape             // après les deux-points on indique la classe
  {                                       // de quelle on produit la héritage
private:
   double         m_radius;               // rayon de cercle

public:
   void           CCircle(){m_type=1;};  // constructeur, le type est égal à 1
   void           SetRadius(double r){m_radius=r;};
   virtual double GetArea(){return (3.14*m_radius*m_radius);}// la surface de cercle
  };
```

Pour le Carré la déclaration de classe est la même:

```
//--- classe dérivée Carré
class CSquare : public CShape             // après les deux-points on indique la classe
```

© 2000-2015, MetaQuotes Software Corp.

```
  {                                        // de quelle on produit la héritage
private:
   double          m_square_side;        // côté de carré

public:
   void            CSquare(){m_type=2;}; // constructeur, le type est égual à 2
   void            SetSide(double s){m_square_side=s;};
   virtual double  GetArea(){return (m_square_side*m_square_side);}//la surface de ca
  };
```

Pour calculer la surface du carré et du cercle on a besoin des valeurs correspondantes des membres m_radius et m_square_side, donc nous avons ajouté les fonctions SetRadius et SetSide() dans la déclaration de la classe correspondante.

Il est supposé que dans le programme on utilise les objets du différent type (CCircle et CSquare), mais hérités d'un type de base CShape. Le polymorphisme nous permet de créer le tableau des objets du type de base CShape, mais les objets eux-mêmes sont encore inconnus à la déclaration de ce tableau et leur type n'est pas défini.

La décision sur l'objet de type de quel sera contenu dans chaque élément du tableau, sera prise directement à l'exécution du programme. Cela sous-entend une création dynamique des objets des classes correspondantes, et donc, la nécessité au lieu des objets eux-mêmes d'utiliser les pointeurs des objets.

Pour la création dynamique des objets on utilise l'opérateur new, il faut indépendamment et évidemment supprimer chaque tel objet par l'opérateur delete. C'est pourquoi nous déclarons le tableau des pointeurs du type CShape et pour son chaque élément nous créons l'objet du type nécessaire (**new Le nom_de la classe**), comme c'est montré dans l'exemple du script:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- déclarons le tableau des pointeurs des objets du type de base
   CShape *shapes[5];   // le tableau des pointeurs sur les objets CShape

//--- remplissons ici le tableau par les objets dérivés
//--- déclarons le pointeur sur l'objet du type CCircle
   CCircle *circle=new CCircle();
//--- spécifions les propriétés de l'objet selon le pointeur circle
   circle.SetRadius(2.5);
//--- plaçons la valeur du pointeur dans shapes[0]
   shapes[0]=circle;

//--- créons encore un objet CCircle et l'enregistrons dans le pointeur shapes[1]
   circle=new CCircle();
   shapes[1]=circle;
   circle.SetRadius(5);

//--- nous avons intentionnellement "oublié" de définir ici la valeur pour shapes[2]
```

© 2000-2015, MetaQuotes Software Corp.

```
//circle=new CCircle();
//circle.SetRadius(10);
//shapes[2]=circle;

//--- définissons la valeur NULL pour l'élément non utilisé
   shapes[2]=NULL;

//--- créons l'objet CSquare et l'enregistrons dans le pointeur shapes[3]
   CSquare *square=new CSquare();
   square.SetSide(5);
   shapes[3]=square;

//--- créons l'objet CSquare et l'enregistrons dans le pointeur shapes[4]
   square=new CSquare();
   square.SetSide(10);
   shapes[4]=square;

//--- il y a le tableau des pointeurs, recevons sa taille
   int total=ArraySize(shapes);
//--- passons dans la boucle selon tous les pointeurs dans le tableau
   for(int i=0; i<5;i++)
     {
      //--- si le pointeur selon l'index spécifié est valide
      if(CheckPointer(shapes[i])!=POINTER_INVALID)
        {
         //--- déduisons le type et la superficie de la figure dans le log
         PrintFormat("L'objet du type %d a la superficie %G",
               shapes[i].GetType(),
               shapes[i].GetArea());
        }
      //--- si le pointeur a le type POINTER_INVALID
      else
        {
         //--- informons de l'erreur
         PrintFormat("L'objet shapes[%d] n'est pas initialisé! Son pointeur %s",
                     i,EnumToString(CheckPointer(shapes[i])));
        }
     }

//--- nous devons indépendamment supprimer tous les objets dynamiques créés
   for(int i=0;i<total;i++)
     {
      //--- on peut supprimer seulement les objets, dont le pointeur a le type POINTE
      if(CheckPointer(shapes[i])==POINTER_DYNAMIC)
        {
         //--- informons de la suppression
         PrintFormat("Supprimons shapes[%d]",i);
         //--- supprimons l'objet selon son pointeur
         delete shapes[i];
```

© 2000-2015, MetaQuotes Software Corp.

```
            }
        }
    }
```

Prêtez attention qu'à la suppression de l'objet par l'opérateurdelete il faut vérifier le type de son pointeur. On peut supprimer seulement les objets qui ont le pointeurPOINTER_DYNAMICà l'aide de delete, et on recevra l'erreur pour les pointeurs de l'autre type.

Mais en plus du fait de redéfinir de fonctions pendant l'héritage, polymorphisme inclut aussi l'implémentation de la même fonction avec de différents ensembles des paramètres dans une classe. Cela signifie que dans la classe on peut définir quelques fonctions avec le même nom, mais avec un différent type et/ou l'ensemble des paramètres. Dans ce cas le polymorphisme se réalise par la surcharge des fonctions.

Voir aussi

Une bibliothèque standard

© 2000-2015, MetaQuotes Software Corp.

La surcharge

Dans une classe il est possible de définir au moins deux méthodes qui utilisent le même nom, mais ont de différents nombres de paramètres. Quand cela a lieu, les méthodes s'appellent *surchargées*, et un processus est appellé *la surcharge de méthode*. La surcharge de la méthode – un des moyens, à l'aide duquel se réalise le polymorphisme. La surcharge des méthodes dans les classes est produite selon les mêmes règles que la surcharge des fonctions.

Si pour la fonction appelée il n'y a pas de conformité exacte, le compilateur produit la recherche de la fonction convenante, sur les trois niveaux successivement:

1. la recherche parmi les méthodes de la classe;
2. la recherche parmi les méthodes des classes de base, systématiquement du prédécesseur le plus proche au premier;
3. la recherche parmi les autres fonctions.

S'il n'y a aucune correspondance exacte à tous les niveaux, mais plusieurs fonctions convenables aux différents niveaux ont été trouvées, on utilise la fonction trouvée au plus petit niveau. Dans un niveau, il ne peut pas y avoir plus qu'une fonction convenable.

A MQL5 il n'y a pas de surcharge des opérateurs.

Voir aussi

Surcharge des fonctions

© 2000-2015, MetaQuotes Software Corp.

Les fonctions virtuelles

Le mot- clé virtual sert du spécificateur de la fonction, qui assure le mécanisme pour le choix dynamique à l'étape de l'exécution de la fonction-membre convenante parmi les fonctions des classes de base et dérivée, les structures ne peuvent pas avoir les fonctions virtuelles. Il peut être utilisé pour changer les déclarations les déclarations pour les membres de la fonction seulement.

La fonction virtuelle, comme une fonction ordinaire, doit avoir un corps exécutable. A l'appel sa sémantique est exactement la même, comme dans les autres fonctions.

La fonction virtuelle peut être remplacée dans la classe dérivée. Le choix quelle définition de fonction appeler pour une fonction virtuelle, est fait dynamiquement (à l'étape de l'exécution). Un cas typique est quand une classe de base contient une fonction virtuelle et les classes dérivées ont leurs propres versions de cette fonction.

Le pointeur sur la classe de base peut indiquer à l'objet de classe, ou à l'objet d'une classe dérivée. Le choix de la fonction-membre appelée sera produit à l'étape de l'exécution et dépendra du type de l'objet, et non du type du pointeur. S'il n'y a aucun membre d'un type dérivé une fonction virtuelle de la classe de base est utilisée par défaut.

Les déstructeurs sont toujours virtuels, indépendamment du fait, sont-ils déclarés avec le mot-clé virtual ou non.

Considérons l'utilisation des fonctions virtuelles à l'exemple du programme MT5_Tetris.mq5. La classe de base CTetrisShape avec une fonction virtuelle Draw (dessiner) est définie dans le fichier inclus MT5_TetisShape.mqh .

```
//+------------------------------------------------------------------+
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
   int               GetType()                  { return(m_type);        }
   void              Left()                     { m_xpos-=SHAPE_SIZE;    }
   void              Right()                    { m_xpos+=SHAPE_SIZE;    }
```

© 2000-2015, MetaQuotes Software Corp.

```
   void              Rotate()                    { m_prev_turn=m_turn; if(++m_turn>3) m
   virtual void      Draw()                      { return;                 }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
  };
```

Ensuite pour chaque classe dérivée cette fonction est réalisée conformément aux particularités de la classe-descendante. Par exemple, la première figure CTetrisShape1 a son implémentation de fonction Draw():

```
class CTetrisShape1 : public CTetrisShape
  {
public:
   //--- dessin de la figure
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      if(m_turn==0 || m_turn==2)
        {
         //--- bâton horizontal
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
           }
        }
      else
        {
         //--- bâton vertical
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+i*SHAPE_SIZE);
           }
        }
     }
  }
```

La figure le carré est décrit par la classe CTetrisShape6 et elle a sa propre implémentation de méthode Draw():

```
class CTetrisShape6 : public CTetrisShape
  {
public:
   //--- dessin de la figure
```

© 2000-2015, MetaQuotes Software Corp.

```
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      for(i=0; i<2; i++)
        {
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
        }
      for(i=2; i<4; i++)
        {
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+(i-2)*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+SHAPE_SIZE);
        }
     }
  };
```

En fonction de celui-là, l'objet de quelle classe est créé, une fonction virtuelle de n'importe quelle classe dérivée est appelée.

```
void CTetrisField::NewShape()
  {
//--- nous créons une de 7 figures possibles par hasard

   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- dessinons
   m_shape.Draw();
//---
  }
```

Voir aussi

Une bibliothèque standard

© 2000-2015, MetaQuotes Software Corp.

Les membres statiques de la classe/structure

Les membres statiques

Les membres de la classe peuvent être annoncés avec l'utilisation du modificateur de la classe de la mémoirestatic. Tels membres des données se divisent par tous les exemplaires de la classe donnée et se trouvent dans une seule place. Les membres non-statiques des données sont créés pour chaque variable-objet de la classe.

L'absence de la possibilité de déclarer statiquement les membres de la classe amènerait à la nécessité d'annoncer ces données au niveau globaldu programme. Cela romprait les relations entre les données et leur classe, ainsi que cela ne s'accordera pas au paradigme principal de la POO – le groupement des données et les méthodes pour leur traitement dans la classe. Le membre statique permet aux données de la classe, qui ne sont pas spécifiques à l'exemplaire particulier exister dans le domaine de la visibilité de la classe.

Puisque le membre statique de la classe ne dépend pas de l'exemplaire concret, l'appel à lui a l'aspect suivant:

```
class_name::variable
```

où*class_name* – c'est le nom de la classe, et *variable* signifie le nom du membre de la classe.

Comme vous le voyez, pour l'appel au membre statique de la classe on utilise l'opérateur de la permission du contexte ::. A l'appel au membre statique à l'intérieur des méthodes de la classe l'opérateur du contexte n'est pas obligatoire.

Il faut évidemment initialiser le membre statique de la classe par la valeur nécessaire, pour cela il doit être annoncé et initialisé au niveau global. L'ordre de l'initialisation des membres statiques correspondra à l'ordre de leur annonce au niveau global dans le code initial.

Par exemple, nous avons la classe *CParser*, destiné pour l'analyse syntaxique des textes, et il nous faut trouver le nombre total des mots et des symboles traités. Il suffit de déclarer les membres nécessaires de la classe comme statiques et les initialiser au niveau global. Alors tous les exemplaires de la classe utiliseront les compteurs totaux des mots et les symboles au travail.

```
//+------------------------------------------------------------------+
//| La classe "Analyseur des textes"                                 |
//+------------------------------------------------------------------+
class CParser
  {
public:
   static int        s_words;
   static int        s_symbols;
   //---le constructeur et le destructeur
                     CParser(void);
                    ~CParser(void){};
  };
...
//--- l'initialisation des membres statiques de la classe Parser au niveau global
int CParser::s_words=0;
int CParser::s_symbols=0;
```

On peut déclarer le membre statique de la classe avec le mot-clé *const*. Ces constantes statiques doivent être initialisés au niveau global avec le mot-clé *const:*

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| La classe "Pile" pour stocker les données traitées               |
//+------------------------------------------------------------------+
class CStack
  {
public:
                     CStack(void);
                    ~CStack(void){};
...
private:
   static const int  s_max_length; // la capacité maximale de la pile
  };

//--- l'initialisation de la constante statique de la classe CStack
const int CStack::s_max_length=1000;
```

Indicateur this

Le mot-clé this signifie l'indicateur déclaré implicitement à lui-même - au exemplaire concret de la classe, dans le contexte de laquelle la méthode est exécutée. Il peut être utilisé seulement dans les méthodes non statiques de la classe. L'indicateur this est le membre implicite non statique de n'importe quelle classe.

On peut s'adresser dans les fonctions statiques seulement aux membres/méthodes statiques de la classe.

Les méthodes statiques

Il est permis d'utiliser les fonctions-membres du type static dans MQL5. Le modificateur *static* doit être devant le type rendu de la fonction dans la déclaration à l'intérieur de la classe.

```
class CStack
  {
public:
   //--- le constructeur et le destructeur
                     CStack(void){};
                    ~CStack(void){};
   //--- la capacitй maximale de la pile
   static int        Capacity();
private:
   int               m_length;     // le nombre d'йlйments dans la pile
   static const int  s_max_length; // la capacitй maximale de la pile
  };
//+------------------------------------------------------------------+
//| Rend le nombre maximal d'йlйments pour stockage dans la pile     |
//+------------------------------------------------------------------+
int CStack::Capacity(void)
  {
   return(s_max_length);
  }
//--- l'initialisation de la constante statique de la classe CStack
const int CStack::s_max_length=1000;
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- объявим переменную типа CStack
   CStack stack;
//--- вызываем статический метод объекта
   Print("CStack.s_max_length=",stack.Capacity());
//--- а можно и так вызвать, т.к. метод статический и не требует наличия объекта
   Print("CStack.s_max_length=",CStack::Capacity());
  }
```

La méthode avec le modificateur const est permanente et ne peut pas modifier les membres implicites de sa classe. La déclaration des fonctions constantes de la classe et des paramètres constants s'appelle *le contrôle de la constance* (const-correctness). Grâce à un tel contrôle on peut être sûr que le compilateur suivra la constance des valeurs des objets et donnera l'erreur encore à l'étape de la compilation en cas de la violation.

Le modificateurconst est mis après la liste des arguments à l'intérieur de la déclaration de la classe. La définition en dehors de la classe doit insérer aussi le modificateur *const*:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| La classe "Rectangle"                                            |
//+------------------------------------------------------------------+
class CRectangle
  {
private:
   double            m_width;      // la largeur
   double            m_height;     //la hauteur
public:
   //--- les constructeurs et le destructeur
                     CRectangle(void):m_width(0),m_height(0){};
                     CRectangle(const double w,const double h):m_width(w),m_height(h)
                    ~CRectangle(void){};
   //--- le calcul de la surface
   double            Square(void) const;
   static double     Square(const double w,const double h);// { return(w*h); }
  };
//+------------------------------------------------------------------+
//| Rend la surface de l'objet "Rectangle"                           |
//+------------------------------------------------------------------+
double CRectangle::Square(void) const
  {
   return(Square(m_width,m_height));
  }
//+------------------------------------------------------------------+
//| Rend l'oeuvre de deux variables                                  |
//+------------------------------------------------------------------+
static double CRectangle::Square(const double w,const double h)
  {
   return(w*h);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- créons un rectangle rect avec des côtés 5 et 6
   CRectangle rect(5,6);
//---déduisons la surface du rectangle par la méthode constante
   PrintFormat("rect.Square()=%.2f",rect.Square());
//---déduisons l'oeuvre des nombres à l'aide de la méthode statique de la classe CRect
   PrintFormat("CRectangle::Square(2.0,1.5)=%f",CRectangle::Square(2.0,1.5));
  }
```

Un argument supplémentaire en faveur de l'utilisation du contrôle de l'intégrité est le fait que dans ce cas, le compilateur génère une optimisation spéciale, par exemple, dispose un objet constantdans la mémoire seulement pour la lecture.

La fonction statique ne peut pas être définie avec le modificateurconst, puisque le modificateur donné garantit la constance des membres de l'exemplaire à l'appel d'une telle fonction. Mais, comme disait déjà plus haut, la fonction statique par définition ne peut pas s'adresser aux membres non statiques de la classe.

Voir aussi

Les variables statiques, Les variables, Les liens. Le modificateur & et le mot clé this

© 2000-2015, MetaQuotes Software Corp.

Les modèles des fonctions

Les fonctions surchargéessont utilisées d'habitude pour l'exécution des opérations semblables sur les différents types de données. L'exemple simple d'une telle fonction dans MQL5 - ArraySize(), qui rend la taille d'un tableau de n'importe quel type. En fait cette fonction de systéme est surchargée, et toute la réalisation d'une telle surcharge est cachée du développeur des logiciels sur MQL5:

```
int  ArraySize(
   void&  array[]      // le tableau vérifiable
   );
```

C'est-à-dire en fait le compilateur de la langue MQL5 met pour chaque appel de cette fonction la réalisation nécessaire, par exemple, pour les tableaux du type entier comme ça:

```
int  ArraySize(
   int&  array[]      // le tableau avec des éléments du type int
   );
```

Et pour le tableau du typeMqlRatespour le travail avec les cotations dans le format des données historiques on peut présenter la fonctionArraySize()ainsi:

```
int  ArraySize(
   MqlRates&  array[] // le tableau rempli par les valeurs du type MqlRates
   );
```

Ainsi, il est très confortable d'utiliser la même fonction pour le travail avec de différents types, mais il est nécessaire de faire tout le travail préalable, faire la surchargede la fonction nécessaire pour tous les types de données, avec lesquelles elle doit travailler correctement.

Il y a une décision plus confortable - si pour chaque type de données doivent être effectuées les opérations identiques, la décision plus compacte et confortable est l'utilisation des modèles des fonctions. Dans ce cas, le programmeur doit écrire une seule description du modèle de la fonction. A une telle description du modèle à titre du paramètre il suffit d'indiquer un certain paramètre formel et pas le type de données non concret, avec lesquelles doit travailler la fonction. Sur la base des types des arguments utilisés lors de l'appel de cette fonction, le compilateur va générer automatiquement les différentes fonctions pour le traitement approprié du chaque type.

La définition du modèle de la fonction commence par le mot-clétemplate,après qui est la liste de paramètres formels en chevrons. Chaque paramètre formel est précédée par le mot-clé typename. Les types formels des paramètres - les types insérés ou les types définis par l'utilisateur. Ils sont utilisés:

- à définir les types des arguments de la fonction
- pour définir le type de la valeur de retour de la fonction,
- pour déclarer des variables à l'intérieur du corps de la description de la fonction

Le nombre de paramètres dans un modèle ne peut pas être plus que huit. Chaque paramètre formel dans la définition du modèle doit quand même une fois apparaître dans la liste des paramètres de la fonction. Chaque nom du paramètre formel doit être unique.

Voici l'exemple du modèle de la fonction pour la recherche de la valeur maximale dans le tableau de n'importe quel type numérique (les nombres entiers et matériels) :

© 2000-2015, MetaQuotes Software Corp.

```
template<typename T>
T ArrayMax(T &arr[])
  {
   uint size=ArraySize(arr);
   if(size==0) return(0);

   T max=arr[0];
   for(uint n=1;n<size;n++)
      if(max<arr[n]) max=arr[n];
//---
   return(max);
  }
```

Ce modèle décrit la fonction qui trouve la valeur maximale dans le tableau transmis et le rend (la valeur trouvée maximale) à titre du résultat. Rappelons que la fonction intégréeArrayMaximum()dans MQL5 rend seulement l'indice de la valeur trouvée maximale, selon lequel l'utilisateur peut recevoir par la suite la valeur elle-même. Par exemple:

```
//--- créons le tableau
   double array[];
   int size=50;
   ArrayResize(array,size);
//--- remplissons par les valeurs accidentelles
   for(int i=0;i<size;i++)
     {
      array[i]=MathRand();
     }

//--- trouvons la position de l'élément maximal dans le tableau
   int max_position=ArrayMaximum(array);
//---maintenant recevons la valeur maximale dans le tableau
   double max=array[max_position];
//--- la sortie de la valeur trouvée
   Print("Max value = ",max);
```

Ainsi, il a fallu deux actions pour recevoir la valeur maximale dans le tableau. A l'aide du modèle de la fonction ArrayMax () on peut recevoir le résultat du type nécessaire, simplement ayant y transmis le tableau du type correspondant. C'est-à-dire au lieu de deux dernières chaînes

```
//--- trouvons la position de l'élément maximal dans le tableau
   int max_position=ArrayMaximum(array);
//--- maintenant recevons la valeur maximale dans le tableau
   double max=array[max_position];
```

maintenant on peut utiliser une chaîne qui rend à la fois le résultat du même type que le tableau transmis:

```
//--- trouvons la valeur maximale
   double max=ArrayMax(array);
```

En cela le type du résultat rendu par la fonction ArrayMax(), correspondra automatiquement au type du tableau.

Pour la création des procédés universels avec différents types de données il est nécessaire d'utiliser le mot-clé typenamepour obtenir le type de l'argument comme la chaîne. Montrons cela à l'exemple de la fonction, qui rend en forme de la chaîne le type de données:

© 2000-2015, MetaQuotes Software Corp.

```
#include <Trade\Trade.mqh>
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CTrade trade;
   double d_value=M_PI;
   int i_value=INT_MAX;
   Print("d_value: type=",GetTypeName(d_value), ",   value=", d_value);
   Print("i_value: type=",GetTypeName(i_value), ",   value=", i_value);
   Print("trade: type=",GetTypeName(trade));
//---
  }
//+------------------------------------------------------------------+
//| Rend le type dans l'aspect de chaîne                             |
//+------------------------------------------------------------------+
template<typename T>
string GetTypeName(const T &t)
  {
//--- rendons le type en forme de la chaîne
   return(typename(T));
//---
  }
```

On peut aussi utiliser les modèles des fonctions aussi pour les méthodes de la classe, par exemple:

```
class CFile
  {
   ...
public:
   ...
   template<typename T>
   uint WriteStruct(T &data);
  };

template<typename T>
uint CFile::WriteStruct(T &data)
  {
   ...
   return(FileWriteStruct(m_handle,data));
  }
```

On ne peut pas annoncer les modèles des fonctions avec les mots clefsexport, virtual et #import.

© 2000-2015, MetaQuotes Software Corp.

Les constantes standards, les énumérations et les structures

Pour simplifier l'écriture de programme et rendre des textes du programme plus convenables pour la perception la langage MQL5 fournit les constantes standards prédéterminées et les énumérations. En outre, pour stocker les informations, on utilise les structures de service.

Les constantes standard sont l'analogue des macrosubstitutions et ont le type int.

Les constantes sont groupées à sa destination:

- Les constantes de graphiques sont utilisées au fonctionnement des graphiques de prix: l'ouverture, la navigation, l'installation des paramètres;
- Les constantes des objetssont destinées au traitement des objets graphiques, qu'on peut créer et afficher sur les graphiques;
- Les constantes des indicateurs servent pour le travail avec les indicateurs standard et d'utilisateur;
- Etat d'environnement – décrivent les propriétés du programme mql5, accordent l'information sur le terminal de client, l'instrument commercial et le compte courant commercial;
- Les constantes commerciales permettent de préciser une information diverse en train du commerce;
- Les constantes nommées sont les constantes du langage MQL5;
- Les structures des données décrivent les formats utilisés du stockage des données;
- Les codes des erreurs et des préventions décrivent les messages du compilateur et le message du serveur commercial sur les requêtes commerciales;
- Les constantes de l'entrée/sortie sont destinées au travail avec les fonctions du fichier et la sortie des messages sur l'écran de l'ordinateur par la fonction MessageBox().

© 2000-2015, MetaQuotes Software Corp.

Les constantes des graphiques

Les constantes décrivant les diverses propriétés des graphiques, sont divisées en groupes suivants:

- Les types des événements – les événements, qui se produisent au fonctionnement du graphique;
- Les périodes des graphiques – les périodes standards insérées;
- Les propriétés des graphiques – les identificateurs utilisés à titre des paramètres des fonctions pour le travail avec des graphiques;
- Les constantes du positionnement - les valeurs du paramètre de la fonction ChartNavigate();
- L'affichage des graphiques - la spécification de l'apparence de graphique.

© 2000-2015, MetaQuotes Software Corp.

Les types des événements de graphique

Il y a 9 types d'événements qui peuvent être traités en utilisant la fonction prédéterminée OnChartEvent(). Pour les événements d'utilisateur on prévoit 65535 identificateurs dans le domain de CHARTEVENT_CUSTOM jusqu'à CHARTEVENT_CUSTOM_LAST inclusivement. Pour la génération de l'événement d'utilisateur il est nécessaire d'utiliser la fonction EventChartCustom().

ENUM_CHART_EVENT

Identificateur	Description
CHARTEVENT_KEYDOWN	Préssion de clavier
CHARTEVENT_OBJECT_CREATE	La création de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_DELETE=true)
CHARTEVENT_OBJECT_CHANGE	L'événement du changement des propriétés de l'objet graphique par le dialogue des propriétés
CHARTEVENT_OBJECT_DELETE	L'effacement de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_DELETE=true)
CHARTEVENT_CLICK	Cliquer sur le graphique
CHARTEVENT_OBJECT_CLICK	Cliquer sur un objet graphique
CHARTEVENT_OBJECT_DRAG	Le glisser-déposer de l'objet graphique
CHARTEVENT_OBJECT_ENDEDIT	La fin de l'édition du texte dans l'objet graphique Edit
CHARTEVENT_CHART_CHANGE	Le changement des dimensions du graphique ou le changement des propriétés du graphique par le dialogue des propriétés
CHARTEVENT_CUSTOM	Le numéro initial de l'événement du domaine des événements d'utilisateur
CHARTEVENT_CUSTOM_LAST	Le numéro final de l'événement du domaine des événements d'utilisateur

Pour chaque type de l'événement les paramètres d'entrée de la fonction OnChartEvent() ont les valeurs définies, qui sont nécessaires au traitement de cet événement. Dans le tableau sont énumérés les événements et les valeurs, qui sont transmises par les paramètres.

L'événement	La valeur du paramètre id	La valeur du paramètre lparam	La valeur du paramètre dparam	La valeur du paramètre sparam
L'événement de la pression du clavier	CHARTEVENT_KEYDOWN	le code de la touche appuyée	Le nombre de pressions du bouton générées au cours de sa	La valeur de ligne d'un masque de bit, décrivant le

© 2000-2015, MetaQuotes Software Corp.

			rétention dans l'état appuyé	statut des boutons du clavier
L'événement de la création de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_CREATE=true)	CHARTEVENT_OBJECT_CREATE	—	—	Le nom de l'objet créé graphique
L'événement du changement des propriétés de l'objet par le dialogue des propriétés	CHARTEVENT_OBJECT_CHANGE	—	—	Le nom de l'objet créé graphique
L'événement de la suppression de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_DELETE=true)	CHARTEVENT_OBJECT_DELETE	—	—	Le nom de l'objet graphique supprimé
L'événement du clic de la souris sur le graphique	CHARTEVENT_CLICK	La coordonnée X	La coordonnée Y	—
L'événement du clic de la souris sur l'objet graphique	CHARTEVENT_OBJECT_CLICK	La coordonnée X	La coordonnée Y	Le nom d'un objet graphique sur lequel l'événement s'est produit
L'événement du déplacement de l'objet graphique à l'aide de la souris	CHARTEVENT_OBJECT_DRAG	—	—	Le nom de l'objet graphique déplacé
L'événement de la fin de l'édition du texte dans l'objet graphique "Le champ de l'entrée"	CHARTEVENT_OBJECT_ENDEDIT	—	—	Le nom de l'objet graphique "Le champ de l'entrée", où l'édition du texte s'est achevée

© 2000-2015, MetaQuotes Software Corp.

L'événement du changement des dimensions du graphique ou le changement des propriétés du graphique par le dialogue des propriétés	CHARTEVENT_CHART_CHANGE	—	—	—
L'événement d'utilisateur avec le numéro N	CHARTEVENT_CUSTOM+N	La valeur, spécifiée par la fonction EventChartCustom()	La valeur, spécifiée par la fonction EventChartCustom()	La valeur, spécifiée par la fonction EventChartCustom()

Exemple:

```
#define KEY_NUMPAD_5       12
#define KEY_LEFT           37
#define KEY_UP             38
#define KEY_RIGHT          39
#define KEY_DOWN           40
#define KEY_NUMLOCK_DOWN   98
#define KEY_NUMLOCK_LEFT  100
#define KEY_NUMLOCK_5     101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP    104
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("Expert avec le nom est lancé ",MQL5InfoString(MQL5_PROGRAM_NAME));
//--- enable object create events
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- enable object delete events
```

© 2000-2015, MetaQuotes Software Corp.

```
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // identificateur de l'événement
                  const long& lparam,   // paramètre de l'événement du type long
                  const double& dparam, // paramètre de l'événement du type double
                  const string& sparam  // paramètre de l'événement du type string
                  )
  {
//--- clic gauche sur le graphique
   if(id==CHARTEVENT_CLICK)
     {
      Print("coordonnées du clic de la souris sur le graphique: x = ",lparam,"  y = ",
     }
//--- clic sur l'objet graphique
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("Clic de la souris sur l'objet avec le nom '"+sparam+"'");
     }
//--- frappe de touche sur le clavier
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch(lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("Est appuyée KEY_NUMLOCK_LEFT");   break;
         case KEY_LEFT:          Print("Est appuyée KEY_LEFT");           break;
         case KEY_NUMLOCK_UP:    Print("Est appuyée KEY_NUMLOCK_UP");     break;
         case KEY_UP:            Print("Est appuyée KEY_UP");             break;
         case KEY_NUMLOCK_RIGHT: Print("Est appuyée KEY_NUMLOCK_RIGHT");  break;
         case KEY_RIGHT:         Print("Est appuyée KEY_RIGHT");          break;
         case KEY_NUMLOCK_DOWN:  Print("Est appuyée KEY_NUMLOCK_DOWN");   break;
         case KEY_DOWN:          Print("Est appuyée KEY_DOWN");           break;
         case KEY_NUMPAD_5:      Print("Est appuyée KEY_NUMPAD_5");       break;
         case KEY_NUMLOCK_5:     Print("Est appuyée KEY_NUMLOCK_5");      break;
         default:                Print("On a appuié quelque touche non énumérée");
        }
      ChartRedraw();
     }
//--- objet est supprimé
   if(id==CHARTEVENT_OBJECT_DELETE)
     {
      Print("Objet avec le nom est supprimé ",sparam);
     }
//--- objet a crée
   if(id==CHARTEVENT_OBJECT_CREATE)
     {
      Print("Objet avec le nom a été crée ",sparam);
     }
//--- objet est déplacé ou les coordonnées des points du rattachement ont modifié
   if(id==CHARTEVENT_OBJECT_DRAG)
     {
      Print("Le changement des points du rattachement de l'objet avec le nom ",sparam)
     }
//--- texte dans la zone de l'entréé de l'objet graphique Edit a modifié
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
     {
      Print("Texte dans l'objet Edit a modifié ",sparam);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
  }
```

Для события CHARTEVENT_MOUSE_MOVE строковой параметр **sparam** содержит число, представляющее информацию о состоянии клавиш:

Бит	Описание
1	Состояние левой клавиши мыши
2	Состояние правой клавиши мыши
3	Состояние клавиши SHIFT
4	Состояние клавиши CTRL
5	Состояние средней клавиши мыши
6	Состояние первой дополнительной клавиши мыши
7	Состояние второй дополнительной клавиши мыши

Пример:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- включение сообщений о перемещении мыши по окну чарта
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse left
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse right
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse middle
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // mouse first X key
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // mouse second X key
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift key
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control key
   return(res);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
```

© 2000-2015, MetaQuotes Software Corp.

```
{
 if(id==CHARTEVENT_MOUSE_MOVE)
    Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
}
```

Voir aussi

Les fonctions du traitement des événements, Le travail avec les événements

© 2000-2015, MetaQuotes Software Corp.

Les périodes des graphiques

Toutes les périodes prédéterminées des graphiques ont les identificateurs uniques. L'identificateur PERIOD_CURRENT signifie une période actuelle du graphique, sur laquelle le programme mql5 est lancé.

ENUM_TIMEFRAMES

Identificateur	Description
PERIOD_CURRENT	Période actuelle
PERIOD_M1	1 minute
PERIOD_M2	2 minutes
PERIOD_M3	3 minutes
PERIOD_M4	4 minutes
PERIOD_M5	5 minutes
PERIOD_M6	6 minutes
PERIOD_M10	10 minutes
PERIOD_M12	12 minutes
PERIOD_M15	15 minutes
PERIOD_M20	20 minutes
PERIOD_M30	30 minutes
PERIOD_H1	1 heure
PERIOD_H2	2 heures
PERIOD_H3	3 heures
PERIOD_H4	4 heures
PERIOD_H6	6 heures
PERIOD_H8	8 heures
PERIOD_H12	12 heures
PERIOD_D1	1 jour
PERIOD_W1	1 semaine
PERIOD_MN1	1 mois

Exemple:

```
   string chart_name="test_Object_Chart";
   Print("essayons de créer l'objet Chart avec le nom ",chart_name);
//--- si un tel objet n'existe pas - nous le créons
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(ObjectFind(0,chart_name)<0)ObjectCreate(0,chart_name,OBJ_CHART,0,0,0,0,0);
//--- spécifions le caractère
   ObjectSetString(0,chart_name,OBJPROP_SYMBOL,"EURUSD");
//--- spécifions la coordonnée X pour le point du rattachement
   ObjectSetInteger(0,chart_name,OBJPROP_XDISTANCE,100);
//--- spécifions la coordonnée Y pour le point du rattachement
   ObjectSetInteger(0,chart_name,OBJPROP_YDISTANCE,100);
//--- établissons la largeur
   ObjectSetInteger(0,chart_name,OBJPROP_XSIZE,400);
//--- établissons la hauteur
   ObjectSetInteger(0,chart_name,OBJPROP_YSIZE,300);
//--- établissons la période
   ObjectSetInteger(0,chart_name,OBJPROP_PERIOD,PERIOD_D1);
//--- établissons l'échelle ( du 0 jusqu'au 5)
   ObjectSetDouble(0,chart_name,OBJPROP_SCALE,4);
//--- Désactivons la sélection par la souris
   ObjectSetInteger(0,chart_name,OBJPROP_SELECTABLE,false);
```

Voir aussi

PeriodSeconds, Period, Date et temps, La visibilité des objets

© 2000-2015, MetaQuotes Software Corp.

Les propriétés des graphiques

Les identificateurs de la famille des énumérations ENUM_CHART_PROPERTY sont utilisés à titre des paramètres des fonctions pour le travail avec les graphiques. L'abréviation r/o dans la colonne "Le type de la propriété" signifie que cette propriété est destinée seulement à la lecture et ne peut pas être changé. L'abréviation w/o dans la colonne "Type de la propriété" signifie que cette propriété est destinée seulement à l'enregistrement et ne peut pas être reçue. A l'appel aux certaines propriétés il est nécessaire d'indiquer le paramètre-modificateur supplémentaire (modifier), qui sert à indiquer le numéro de la sous-fenêtre du graphique. 0 signifie une fenêtre principale.

Les fonctions établissant les propriétés du graphique, servent en fait pour envoyer à lui des commandes pour le changement. À l'exécution réussie de ces fonctions la commande se trouve dans une file d'attente des événements du graphique. Le changement du graphique est produit en train du traitement de la file d'attente des événements du graphique donné.

Pour cette raison il ne faut pas attendre la mise à jour visuelle immédiate du graphique après l'appel des fonctions données. En général, la mise à jour du graphique est produite par le terminal automatiquement selon les événements du changement - l'entrée de la nouvelle cotation, le changement de la taille de la fenêtre du graphique etc.

Pour la mise à jour forcée de l'apparence du graphique utilisez la commande pour redessiner le graphiqueChartRedraw().

Pour les fonctions ChartSetInteger() et ChartGetInteger()

ENUM_CHART_PROPERTY_INTEGER

Identificateur	Description	Type de la propriété
CHART_IS_OBJECT	Le critère pour l'identification de l'objet "Graphique" (OBJ_CHART) - rend true pour l'objet graphique. Pour un vrai graphique la valeur est égale au false	bool r/o
CHART_BRING_TO_TOP	L'affichage du graphique au-dessus de tous les autres	bool w/o
CHART_MOUSE_SCROLL	Le défilement du graphique par un bouton gauche de la souris à l'horizontale. Le défilement selon la verticale sera aussi accessible, si dans true est établi une valeur de l'une des trois propriétés: CHART_SCALEFIX, CHART_SCALEFIX_11 ou CHART_SCALE_PT_PER_BAR	bool
CHART_EVENT_MOUSE_MOVE	L'expédition des messages à tous les programmes mql5 sur	bool

© 2000-2015, MetaQuotes Software Corp.

	le graphique sur les événements du déplacement et les pressions des boutons de la souris (CHARTEVENT_MOUSE_MOVE)	
CHART_EVENT_OBJECT_CREATE	L'expédition des messages à tous les programmes mql5 sur le graphique sur l'événement de la création de l'objet graphique (CHARTEVENT_OBJECT_CREATE)	bool
CHART_EVENT_OBJECT_DELETE	L'expédition des messages à tous les programmes mql5 sur le graphique sur l'événement de la destruction de l'objet graphique (CHARTEVENT_OBJECT_DELETE)	bool
CHART_MODE	Le type de graphique (les bougies, les barres ou la ligne)	enum ENUM_CHART_MODE
CHART_FOREGROUND	Le graphique des prix dans l'arrière-plan	bool
CHART_SHIFT	Le mode de l'alinéa du bord droit du graphique de prix	bool
CHART_AUTOSCROLL	Le mode du déplacement automatique vers le bord droit du graphique	bool
CHART_SCALE	Echelle	int de 0 à 5
CHART_SCALEFIX	Le mode de l'échelle fixée	bool
CHART_SCALEFIX_11	Le mode de l'échelle 1:1	bool
CHART_SCALE_PT_PER_BAR	Le mode de l'indication de l'échelle dans les points sur la barre	bool
CHART_SHOW_OHLC	L'affichage dans un gauche angle supérieur des significations OHLC	bool
CHART_SHOW_BID_LINE	L'affichage de la signification Bid par la ligne horizontale sur le graphique	bool
CHART_SHOW_ASK_LINE	L'affichage de la signification Ask par la ligne horizontale sur le graphique	bool

© 2000-2015, MetaQuotes Software Corp.

CHART_SHOW_LAST_LINE	L'affichage de la signification Last par la ligne horizontale sur le graphique	bool
CHART_SHOW_PERIOD_SEP	L'affichage des délimiteurs verticaux entre les périodes voisines	bool
CHART_SHOW_GRID	L'affichage de la grille dans le graphique	bool
CHART_SHOW_VOLUMES	L'affichage des volumes sur le graphique	enum ENUM_CHART_VOLUME_MODE
CHART_SHOW_OBJECT_DESCR	Les descriptions surgissantes des objets graphiques	bool
CHART_VISIBLE_BARS	Le nombre de barres sur le graphique, accessible pour l'affichage	int r/o
CHART_WINDOWS_TOTAL	Le total des fenêtres du graphique, y compris les sous - fenêtres des indicateurs	int r/o
CHART_WINDOW_IS_VISIBLE	La visibilité des sous - fenêtres	bool r/o Le modificateur - le numéro de sous -fenêtre
CHART_WINDOW_HANDLE	Le handle de graphique (HWND)	int r/o
CHART_WINDOW_YDISTANCE	La distance en pixels selon l'axe vertical Y entre le cadre supérieur de la sous-fenêtre de l'indicateur et le cadre supérieur de la fenêtre principale du graphique. A l'arrivée des événements de la souris les coordonnées du curseur sont transmises aux coordonnées de la fenêtre principale du graphique, pendant que les coordonnées des objets graphiques dans une sous-fenêtre de l'indicateur sont spécifiées relativement un angle gauche supérieur de la sous-fenêtre. La valeur est demandée pour la transfert des coordonnées absolues du graphique principal aux coordonnées locales de la sous-fenêtre pour le travail correct avec les	int r/o Le modificateur - le numéro de sous -fenêtre

© 2000-2015, MetaQuotes Software Corp.

	objets graphiques, dont les coordonnées sont spécifiées relativement un angle gauche supérieur du cadre de la sous-fenêtre.	
CHART_FIRST_VISIBLE_BAR	Le numéro de la première barre visible sur le graphique. L'indexation des barres correspond à la série temporelle.	int r/o
CHART_WIDTH_IN_BARS	La largeur du graphique dans les barres	int r/o
CHART_WIDTH_IN_PIXELS	La largeur du graphique en pixels	int r/o
CHART_HEIGHT_IN_PIXELS	La hauteur du graphique en pixels	int Le modificateur - le numéro de sous -fenêtre
CHART_COLOR_BACKGROUND	La couleur du fond du graphique	color
CHART_COLOR_FOREGROUND	La couleur des axes, les échelles et les lignes OHLC	color
CHART_COLOR_GRID	La couleur de la grille	color
CHART_COLOR_VOLUME	La couleur des volumes et des niveaux de l'ouverture des positions	color
CHART_COLOR_CHART_UP	La couleur de la barre en haut, de l'ombre et de l'encadrement du corps de la bougie d'haussier	color
CHART_COLOR_CHART_DOWN	La couleur de la barre en bas, de l'ombre et de l'encadrement du corps de la bougie de baissier	color
CHART_COLOR_CHART_LINE	La couleur de la ligne du graphique et des chandeliers japonais "Doji"	color
CHART_COLOR_CANDLE_BULL	La couleur du corps de la bougie d'haussier	color
CHART_COLOR_CANDLE_BEAR	La couleur du corps de la bougie de baissier	color
CHART_COLOR_BID	La couleur de la ligne du prix Bid	color
CHART_COLOR_ASK	La couleur de la ligne du prix	color

© 2000-2015, MetaQuotes Software Corp.

	Ask	
CHART_COLOR_LAST	La couleur de la ligne du prix du dernier marché passé (Last)	color
CHART_COLOR_STOP_LEVEL	La couleur des niveaux des ordres Stop (Stop Loss et Take Profit)	color
CHART_SHOW_TRADE_LEVELS	L'affichage sur le graphique des niveaux commerciaux (les niveaux des positions ouvertes, Stop Loss, Take Profit et des ordres remis)	bool
CHART_DRAG_TRADE_LEVELS	L'autorisation de faire glisser les niveaux commerciaux sur le graphique à l'aide de la souris. Par défaut, le mode glisser-déposer est activé (la valeur true)	bool
CHART_SHOW_DATE_SCALE	L'affichage de l'échelle du temps sur le graphique	bool
CHART_SHOW_PRICE_SCALE	L'affichage de l'échelle de prix sur le graphique	bool
CHART_SHOW_ONE_CLICK	Отображение на графике панели быстрой торговли (опция "Торговля одним кликом")	bool

Pour les fonctions ChartSetDouble() et ChartGetDouble()

ENUM_CHART_PROPERTY_DOUBLE

Identificateur	Description	Type de la propriété
CHART_SHIFT_SIZE	Le montant de l'alinéa de la barre nulle du bord droit en pourcentage	double (de 10 jusqu'au 50 pour cent)
CHART_FIXED_POSITION	La situation de la position fixée du graphique du bout gauche en pourcentage. La position fixée du graphique est désignée par un petit triangle gris sur l'axe horizontal du temps et est montrée seulement dans le cas où on déconnecte le défilement automatique vers le bord droit à l'entrée du	double

© 2000-2015, MetaQuotes Software Corp.

	nouveau tick (regarde la propriété CHART_AUTOSCROLL). La barre, qui se trouve sur la position fixée, reste à la même place à l'augmentation et la réduction de l'échelle.	
CHART_FIXED_MAX	Le maximum fixé du graphique	double
CHART_FIXED_MIN	Le minimum fixé du graphique	double
CHART_POINTS_PER_BAR	La signification de l'échelle dans les points sur la barre	double
CHART_PRICE_MIN	Le minimum du graphique	double r/o le modificateur - le numéro de sous -fenêtre
CHART_PRICE_MAX	Le maximum du graphique	double r/o le modificateur - le numéro de sous -fenêtre

Pour les fonctions ChartSetSring() et ChartGetString()

ENUM_CHART_PROPERTY_STRING

Identificateur	Description	Type de la propriété
CHART_COMMENT	Le texte du commentaire sur le graphique	string

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
int chartMode=ChartGetInteger(0,CHART_MODE);
switch(chartMode)
  {
   case(CHART_BARS):    Print("CHART_BARS");   break;
   case(CHART_CANDLES): Print("CHART_CANDLES");break;
   default:Print("CHART_LINE");
  }
bool shifted=ChartGetInteger(0,CHART_SHIFT);
if(shifted) Print("CHART_SHIFT = true");
else Print("CHART_SHIFT = false");
bool autoscroll=ChartGetInteger(0,CHART_AUTOSCROLL);
if(autoscroll) Print("CHART_AUTOSCROLL = true");
else Print("CHART_AUTOSCROLL = false");
int chartHandle=ChartGetInteger(0,CHART_WINDOW_HANDLE);
Print("CHART_WINDOW_HANDLE = ",chartHandle);
int windows=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
Print("CHART_WINDOWS_TOTAL = ",windows);
if(windows>1)
  {
   for(int i=0;i<windows;i++)
     {
      int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,i);
      double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,i);
      double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,i);
      Print(i+": CHART_HEIGHT_IN_PIXELS = ",height," pixels");
      Print(i+": CHART_PRICE_MIN = ",priceMin);
      Print(i+": CHART_PRICE_MAX = ",priceMax);
     }
  }
```

Voir aussi

Les exemples du travail avec le graphique

© 2000-2015, MetaQuotes Software Corp.

Le positionnement du graphique

Trois identificateurs de la liste ENUM_CHART_POSITION ont les valeurs possibles du paramètre position pour la fonction ChartNavigate().

ENUM_CHART_POSITION

Identificateur	Description
CHART_BEGIN	Le début du graphique (les prix les plus vieux)
CHART_CURRENT_POS	La position courante
CHART_END	La fin du graphique (les prix les plus derniers)

Exemple:

```
long handle=ChartOpen("EURUSD",PERIOD_H12);
if(handle!=0)
  {
   ChartSetInteger(handle,CHART_AUTOSCROLL,false);
   ChartSetInteger(handle,CHART_SHIFT,true);
   ChartSetInteger(handle,CHART_MODE,CHART_LINE);
   ResetLastError();
   bool res=ChartNavigate(handle,CHART_END,150);
   if(!res) Print("Navigate failed. Error = ",GetLastError());
   ChartRedraw();
  }
```

© 2000-2015, MetaQuotes Software Corp.

L'affichage des graphiques

On peut afficher les graphiques de prix par trois moyens:

- comme des barres;
- comme des bougies;
- comme la ligne brisée.

Le moyen concret de l'affichage du graphique de prix est spécifié par la fonction ChartSetInteger (handle_du graphique,CHART_MODE, le type_du graphique), où le type_du graphique - une des valeurs de l'énumération ENUM_CHART_MODE.

ENUM_CHART_MODE

Identificateur	Description
CHART_BARS	L'affichage comme les barres
CHART_CANDLES	L'affichage comme les chandeliers japonais
CHART_LINE	L'affichage comme une ligne tracée par les prix Close

Pour l'indication du mode de l'affichage des volumes du graphique de prix on utilise la fonction ChartSetInteger(handle_du graphique, CHART_SHOW_VOLUMES, le type_d'affichage), où le type_d'affichage - une des significations de l'énumération ENUM_CHART_VOLUME_MODE.

ENUM_CHART_VOLUME_MODE

Identificateur	Description
CHART_VOLUME_HIDE	Les volumes ne sont pas montrés
CHART_VOLUME_TICK	Les volumes de tick
CHART_VOLUME_REAL	Les volumes de commerce

Exemple:

```
//--- recevrons handle du graphique actuel
   long handle=ChartID();
   if(handle>0) // si c'est réussi, personnalisons supplémentairement
     {
      //--- désactiverons le défilement automatique
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- établissons l'alinéa du bord droit du graphique
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- affichons comme des bougies
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- faisons défiler sur 100 barres du début de l'histoire
      ChartNavigate(handle,CHART_CURRENT_POS,100);
      //--- établir le mode de l'affichage des volumes de tics
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
     }
```

© 2000-2015, MetaQuotes Software Corp.

Voir aussi

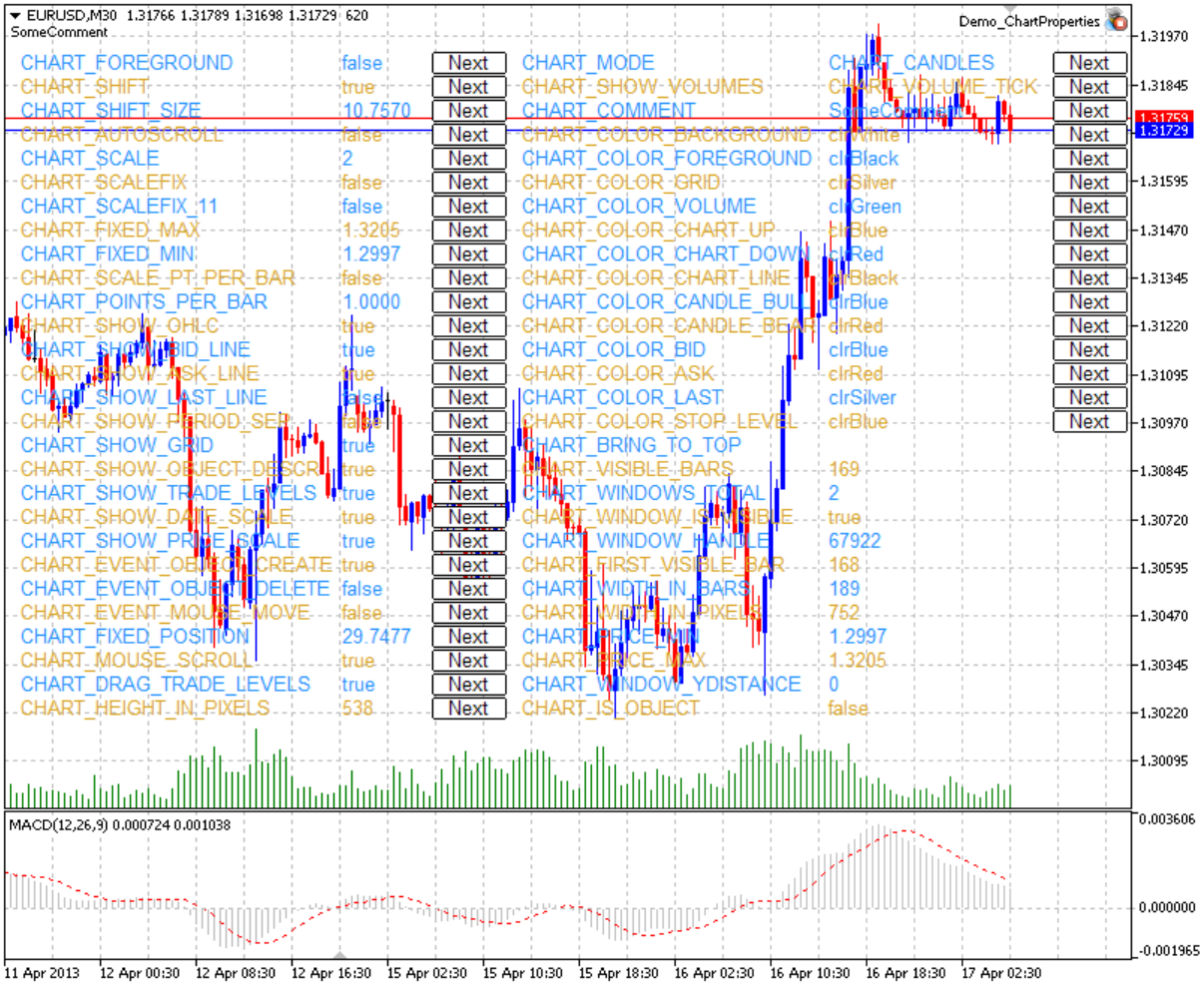
ChartOpen, ChartID

© 2000-2015, MetaQuotes Software Corp.

Les exemples du travail avec le graphique

Dans ce paragraphe on présente les exemples pour le travail avec les propriétés du graphique. Pour chaque propriété sont amenées une ou deux fonctions complètes qui vous permettent de spécifier/ recevoir la valeur de cette propriété. Ces fonctions peuvent être utilisées dans ces programmes MQL5 comme il est.

Sur le dessin est montré le panneau graphique pour illustrer comment le changement de la propriété du graphiquechange sa vue. La pression du bouton "Next" permet d'établir une nouvelle valeur de la propriété correspondante et voir les changements de la fenêtre du graphique.



Le code initial du panneau se trouve plus bas.

Les propriétés du graphique et les exemples des fonctions pour travailler avec eux

- **CHART_IS_OBJECT** -définit, est-ce que l'objet est un vrai graphique oul'objet graphique.

```
//+------------------------------------------------------------------+
//| La définition, est-ce que l'objet est un graphique. Si c'est     |
//| l'objet graphique, le résultat est true. Si c'est un vrai        |
//| graphique, la variable result rendra la valeur false.            |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
bool ChartIsObject(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la propriété du graphique
   if(!ChartGetInteger(chart_ID,CHART_IS_OBJECT,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      //--- rendons false
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_BRING_TO_TOP** - montre le graphique au-dessus de tous les autres.

```
//+------------------------------------------------------------------+
//| L'envoi d'une commande au terminal pour montrer le graphique au-dessus de tous les
//+------------------------------------------------------------------+
bool ChartBringToTop(const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- montrons le graphique au-dessus de tous les autres
   if(!ChartSetInteger(chart_ID,CHART_BRING_TO_TOP,0,true))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_MOUSE_SCROLL** - la propriété du défilement du graphique par le bouton gauche de la souris.

```
//+------------------------------------------------------------------+
//| La fonction définit, si on peut faire défiler le graphique par le bouton gauche |
//| de la souris.                                                    |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool ChartMouseScrollGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------+
//| La fonction active/désactive le défilement du graphique par le bouton gauche |
//| de la souris.                                                                |
//+------------------------------------------------------------------------------+
bool ChartMouseScrollSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_EVENT_MOUSE_MOVE** - la propriété de l'envoi aux programmes mql5 des messages sur les événements du déplacement et des pressions des boutons de la souris (CHARTEVENT_MOUSE_MOVE).

```
//+------------------------------------------------------------------------------+
//| Est-ce qu'on envoie les messages sur les événements du déplacement et de la pressi
//| de la souris à tous les programmes mql5 sur ce graphique?
//+------------------------------------------------------------------------------+
bool ChartEventMouseMoveGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
```

© 2000-2015, MetaQuotes Software Corp.

```
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------------
//| La fonction active/désactive l'envoi des messages sur les événements du déplacemer
//| de la pression des boutons de la souris à tous les programmes mql5 sur ce le graph
//+------------------------------------------------------------------------------------
bool ChartEventMouseMoveSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_EVENT_OBJECT_CREATE** - la propriété de l'expédition aux programmes mql5 des messages sur l'événement de la création de l'objet graphique (CHARTEVENT_OBJECT_CREATE).

```
//+------------------------------------------------------------------------------------
//| Est-ce qu'on envoie les messages sur l'événement de la création de l'objet graphic
//| à tous les programmes mql5 sur ce graphique?
//+------------------------------------------------------------------------------------
bool ChartEventObjectCreateGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive l'envoi de messages sur l'événement |
//| de la création de l'objet graphique à tous les programmes mql5 sur ce |
//| le graphique.                                                    |
//+------------------------------------------------------------------+
bool ChartEventObjectCreateSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_EVENT_OBJECT_DELETE** -la propriété de l'expédition aux programmes mql5 des messages sur l'événement de la suppression de l'objet graphique (CHARTEVENT_OBJECT_DELETE).

```
//+------------------------------------------------------------------+
//| Est-ce qu'on envoie les messages sur l'événement de la suppression |
//| de l'objet graphique à tous les programmes mql5 sur ce graphique?  |
//+------------------------------------------------------------------+
bool ChartEventObjectDeleteGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
```

© 2000-2015, MetaQuotes Software Corp.

```
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------+
//| La fonction active/désactive l'envoi de messages sur l'événement       |
//| de la suppression de l'objet graphique à tous les programmes mql5 sur ce |
//| le graphique.                                                          |
//+------------------------------------------------------------------------+
bool ChartEventObjectDeleteSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_MODE** - le type du graphique (les bougies, les barres ou la ligne).

```
//+------------------------------------------------------------------+
//| La réception du type de l'affichage du graphique                 |
//| (en forme des bougies, des barres ou de la ligne)                |
//+------------------------------------------------------------------+
ENUM_CHART_MODE ChartModeGet(const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long result=WRONG_VALUE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_MODE,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((ENUM_CHART_MODE)result);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| L'établissement du type de l'affichage du graphique              |
//| (en forme des bougies, des barres ou de la ligne)                |
//+------------------------------------------------------------------+
bool ChartModeSet(const long value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_MODE,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_FOREGROUND** - la propriété de l'affichage du graphique de prix au premier plan.

```
//+------------------------------------------------------------------+
//| La fonction définit, si le graphique de prix est affiché         |
//| au premier plan.                                                 |
//+------------------------------------------------------------------+
bool ChartForegroundGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive l'affichage du graphique de prix au |
//| premier plan.                                                    |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool ChartForegroundSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHIFT** - le régime de l'alinéa du graphique de prix du bord droit.

```
//+------------------------------------------------------------------+
//| La fonction définit si le mode de l'affichage du graphique de prix |
//| avec alinéa du bord droit est activé.                            |
//+------------------------------------------------------------------+
bool ChartShiftGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive le mode de l'affichage du graphique |
//| de prix avec alinéa du bord droit.                               |
//+------------------------------------------------------------------+
bool ChartShiftSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!ChartSetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_AUTOSCROLL** - le mode du passage automatique vers le bord droit du graphique.

```
//+------------------------------------------------------------------+
//| La fonction définit si le mode du défilement automatique         |
//| du graphique à droite à l'entrée des nouveaux ticks est activé   |
//+------------------------------------------------------------------+
bool ChartAutoscrollGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive le mode du défilement automatique   |
//| du graphique à droite à l'entrée des nouveaux ticks.             |
//+------------------------------------------------------------------+
bool ChartAutoscrollSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SCALE** - la propriété de l'échelle du graphique.

```
//+------------------------------------------------------------------+
//| La réception de l'échelle du graphique (de 0 jusqu'à 5)          |
//+------------------------------------------------------------------+
int ChartScaleGet(const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long result=-1;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SCALE,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((int)result);
  }
//+------------------------------------------------------------------+
//| L'établissement de l'échelle du graphique (de 0 jusqu'à 5)       |
//+------------------------------------------------------------------+
bool ChartScaleSet(const long value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SCALE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SCALEFIX** - le mode de l'échelle fixée du graphique.

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| La fonction définit si le mode de l'échelle fixée est activé     |
//+------------------------------------------------------------------+
bool ChartScaleFixGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive le mode de l'échelle fixée          |
//+------------------------------------------------------------------+
bool ChartScaleFixSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SCALEFIX_11** - le mode de l'échelle du graphique 1:1.

```
//+------------------------------------------------------------------+
//| La fonction définit si le mode de l'échelle "1:1" est activé     |
//+------------------------------------------------------------------+
bool ChartScaleFix11Get(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive le mode de l'échelle "1:1"          |
//+------------------------------------------------------------------+
bool ChartScaleFix11Set(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SCALE_PT_PER_BAR** - le mode de l'indication de l'échelle du graphique dans les points sur une barre.

```
//+------------------------------------------------------------------+
//| La fonction définit si le mode de l'indication de l'échelle      |
//| dans les points sur une barre est activé                         |
//+------------------------------------------------------------------+
bool ChartScalePerBarGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive de l'indication de l'échelle        |
//| du graphique dans les points sur une barre                       |
//+------------------------------------------------------------------+
bool ChartScalePerBarSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHOW_OHLC** - la propriété de l'affichage dans un angle gauche supérieur des valeurs OHLC.

```
//+------------------------------------------------------------------+
//| La fonction définit si le mode de l'affichage des valeurs OHLC   |
//| dans un angle gauche supérieur est activé.                       |
//+------------------------------------------------------------------+
bool ChartShowOHLCGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
```

© 2000-2015, MetaQuotes Software Corp.

```
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive le mode de l'affichage des          |
//| valeurs OHLC dans un angle gauche supérieur du graphique.        |
//+------------------------------------------------------------------+
bool ChartShowOHLCSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHOW_BID_LINE** - la propriété de l'affichage de la valeur Bid par la ligne horizontale sur le graphique.

```
//+------------------------------------------------------------------+
//| La fonction définit si le mode de l'affichage de                 |
//| la ligne Bid sur le graphique.                                   |
//+------------------------------------------------------------------+
bool ChartShowBidLineGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| La fonction active/désactive l'affichage de la ligne Bid sur     |
//| le graphique.                                                    |
//+------------------------------------------------------------------+
bool ChartShowBidLineSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHOW_ASK_LINE** - la propriété de l'affichage de la valeur Ask par la ligne horizontale sur le graphique.

```
//+------------------------------------------------------------------+
//| La fonction définit si le mode de l'affichage de                 |
//| la ligne Ask sur le graphique.                                   |
//+------------------------------------------------------------------+
bool ChartShowAskLineGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive l'affichage de la ligne Ask sur     |
//| le graphique.                                                    |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool ChartShowAskLineSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHOW_LAST_LINE** - la propriété de l'affichage de la valeur Last par la ligne horizontale sur le graphique.

```
//+------------------------------------------------------------------+
//| La fonction définit si le mode de l'affichage de                 |
//| la ligne pour le prix du dernier marché passé.                   |
//+------------------------------------------------------------------+
bool ChartShowLastLineGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive l'affichage de                      |
//| la ligne du prix du dernier marché passé.                        |
//+------------------------------------------------------------------+
bool ChartShowLastLineSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHOW_PERIOD_SEP** - la propriété de l'affichage des délimiteurs verticals entre les périodes voisines.

```
//+------------------------------------------------------------------+
//| La fonction définit si le mode de l'affichage des délimiteurs    |
//| verticals entre les périodes voisines.                           |
//+------------------------------------------------------------------+
bool ChartShowPeriodSeparatorGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive le mode de l'affichage              |
//| des délimiteurs verticals entre les périodes voisines.           |
//+------------------------------------------------------------------+
bool ChartShowPeriodSepapatorSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
```

© 2000-2015, MetaQuotes Software Corp.

```
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHOW_GRID** - la propriété de l'affichage de la grille sur le graphique.

```
//+------------------------------------------------------------------+
//| La fonction définit, si la grille est affichée sur le graphique  |
//+------------------------------------------------------------------+
bool ChartShowGridGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------+
//| La fonction active/désactive l'affichage de la grille sur le graphique |
//+------------------------------------------------------------------------+
bool ChartShowGridSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

- **CHART_SHOW_VOLUMES** - la propriété de l'affichage des volumes sur le graphique.

```
//+------------------------------------------------------------------------------------+
//| La fonction définit, si les volumes sont affichés sur le graphique (ne sont pas |
//| affichés, sont affichés les volumes de tick, sont affichés les volumes réels)   |
//+------------------------------------------------------------------------------------+
ENUM_CHART_VOLUME_MODE ChartShowVolumesGet(const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long result=WRONG_VALUE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_VOLUMES,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((ENUM_CHART_VOLUME_MODE)result);
  }
//+------------------------------------------------------------------------------+
//| La fonction établit le mode de l'affichage des volumes sur le graphique |
//+------------------------------------------------------------------------------+
bool ChartShowVolumesSet(const long value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_VOLUMES,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHOW_OBJECT_DESCR** - la propriété des descriptions émergeantes des objets graphiques.

```
//+------------------------------------------------------------------+
//| La fonction définit, si les descriptions émergeantes des objets  |
//| graphiques sont affichées à l'induction sur eux par la souris    |
//+------------------------------------------------------------------+
bool ChartShowObjectDescriptionGet(bool &result,const long chart_ID=0)
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------+
//| La fonction active / désactive l'affichage des descriptions émergeant  |
//| des objets graphiques sont affichées à l'induction sur eux par la souris |
//+------------------------------------------------------------------------+
bool ChartShowObjectDescriptionSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_VISIBLE_BARS** - définit le nombre de barres sur le graphique disponibles pour l'affichage.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la quantité de barres, qui s'affichent        |
//| (sont visibles) dans la fenêtre du graphique.                    |
//+------------------------------------------------------------------+
int ChartVisibleBars(const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long result=-1;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_VISIBLE_BARS,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_WINDOWS_TOTAL** - définit le nombre total de fenêtres du graphique, y compris les sous-fenêtres des indicateurs.

```
//+------------------------------------------------------------------+
//| La fonction reçoit le nombre total de fenêtres du graphique,     |
//| y compris les sous-fenêtres des indicateurs.                     |
//+------------------------------------------------------------------+
int ChartWindowsTotal(const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long result=-1;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WINDOWS_TOTAL,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_WINDOW_IS_VISIBLE** - définit la visibilité du sous-fenêtre.

```
//+------------------------------------------------------------------+
//| La fonction définit si cette fenêtre ou                          |
//| le sous-fenêtre du graphique est visible                         |
//+------------------------------------------------------------------+
bool ChartWindowsIsVisible(bool &result,const long chart_ID=0,const int sub_window=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_IS_VISIBLE,sub_window,value))
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_WINDOW_HANDLE** - rend le handle du graphique.

```
//+------------------------------------------------------------------+
//| La fonction reçoit le handle du graphique                        |
//+------------------------------------------------------------------+
int ChartWindowsHandle(const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long result=-1;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_HANDLE,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_WINDOW_YDISTANCE** -définit la distance en pixels entre le cadre supérieur du sous-fenêtre de l'indicateur et le cadre supérieur de la fenêtre principale du graphique.

```
//+------------------------------------------------------------------------+
//| La fonction reçoit la distance en pixels entre le cadre supérieur du   |
//| sous-fenêtre et le cadre supérieur de la fenêtre principale du graphique |
//+------------------------------------------------------------------------+
int ChartWindowsYDistance(const long chart_ID=0,const int sub_window=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long result=-1;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_YDISTANCE,sub_window,result))
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       //---  déduisons le message sur l'erreur au journal "Experts"
       Print(__FUNCTION__+", Error Code = ",GetLastError());
      }
//--- rendons la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_FIRST_VISIBLE_BAR** - rend le numéro du première barre visible sur le graphique (l'indexation des barres correspond à la série temporelle).

```
//+------------------------------------------------------------------------------------
//| La fonction reçoit le numéro de la première barre visible sur le graphique.
//| L'indexation est comme dans une série temporelle, les dernières barres ont des inc
//+------------------------------------------------------------------------------------
int ChartFirstVisibleBar(const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long result=-1;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_FIRST_VISIBLE_BAR,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_WIDTH_IN_BARS** - rend la largeur du graphique en barres.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la valeur de la largeur du graphique en barres |
//+------------------------------------------------------------------+
int ChartWidthInBars(const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long result=-1;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_BARS,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

© 2000-2015, MetaQuotes Software Corp.

```
     }
//--- rendons la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_WIDTH_IN_PIXELS** - rend la largeur du graphique en pixels.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la valeur de la largeur du graphique en pixels |
//+------------------------------------------------------------------+
int ChartWidthInPixels(const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long result=-1;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_PIXELS,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((int)result);
  }
```

- **CHART_HEIGHT_IN_PIXELS** - la propriété de la hauteur du graphique en pixels.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la valeur de la hauteur du graphique en pixels |
//+------------------------------------------------------------------+
int ChartHeightInPixelsGet(const long chart_ID=0,const int sub_window=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long result=-1;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((int)result);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| La fonction établit la valeur de la hauteur du graphique en pixels |
//+------------------------------------------------------------------+
bool ChartHeightInPixelsSet(const int value,const long chart_ID=0,const int sub_windo
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_BACKGROUND** - la couleur du fond du graphique

```
//+------------------------------------------------------------------+
//| La fonction reçoit la couleur du fond du graphique.              |
//+------------------------------------------------------------------+
color ChartBackColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur du fond du graphique
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BACKGROUND,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur du fond du graphique.             |
//+------------------------------------------------------------------+
bool ChartBackColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//---établissons la couleur du fond du graphique
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BACKGROUND,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
```

© 2000-2015, MetaQuotes Software Corp.

```
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_FOREGROUND** - la couleur des axes, de l'échelle et la ligne OHLC.

```
//+------------------------------------------------------------------------------------------+
//| La fonction reçoit la couleur des axes, de l'échelle et la ligne OHLC du graphique
//+------------------------------------------------------------------------------------------+
color ChartForeColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur des axes, de l'échelle et la ligne OHLC du graphique
   if(!ChartGetInteger(chart_ID,CHART_COLOR_FOREGROUND,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------------------------------+
//| La fonction établit la couleur des axes, de l'échelle et la ligne OHLC du graphiqu
//+------------------------------------------------------------------------------------------+
bool ChartForeColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur des axes, de l'échelle et la ligne OHLC du graphique
   if(!ChartSetInteger(chart_ID,CHART_COLOR_FOREGROUND,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_GRID** - la couleur de la grille du graphique.

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la couleur de la grille du graphique          |
//+------------------------------------------------------------------+
color ChartGridColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur de la grille du graphique
   if(!ChartGetInteger(chart_ID,CHART_COLOR_GRID,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur de la grille du graphique         |
//+------------------------------------------------------------------+
bool ChartGridColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur de la grille du graphique
   if(!ChartSetInteger(chart_ID,CHART_COLOR_GRID,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_VOLUME** - la couleur des volumes et des niveaux de l'ouverture des positions.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la couleur de l'affichage des volumes         |
//| et des niveaux de l'ouverture des positions.                     |
//+------------------------------------------------------------------+
color ChartVolumeColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- recevrons la couleur des volumes et des niveaux de l'ouverture des positions
   if(!ChartGetInteger(chart_ID,CHART_COLOR_VOLUME,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur des volumes                       |
//| et des niveaux de l'ouverture des positions.                     |
//+------------------------------------------------------------------+
bool ChartVolumeColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur des volumes et des niveaux de l'ouverture des positions
   if(!ChartSetInteger(chart_ID,CHART_COLOR_VOLUME,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_CHART_UP** - la couleur de la barre en haut, de l'ombre et de l'encadrement du corps de la bougie d'haussier.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la couleur de la barre en haut, la couleur    |
//| de l'ombre et de l'encadrement du corps de la bougie d'haussier  |
//+------------------------------------------------------------------+
color ChartUpColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur de la barre en haut, de l'ombre et de l'encadrement du corp
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_UP,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur de la barre en haut, la couleur   |
//| de l'ombre et de l'encadrement du corps de la bougie d'haussier  |
//+------------------------------------------------------------------+
bool ChartUpColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur de la barre en haut, de l'ombre et de l'encadrement du co
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_UP,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_CHART_DOWN** - la couleur de la barre en bas, de l'ombre et de l'encadrement du corps de la bougie de baissier.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la couleur de la barre en bas, la couleur     |
//| de l'ombre et de l'encadrement du corps de la bougie de baissier |
//+------------------------------------------------------------------+
color ChartDownColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur de la barre en bas, de l'ombre et de l'encadrement du corps
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_DOWN,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur de la barre en bas, de l'ombre et |
//| de l'encadrement du corps de la bougie de baissier               |
//+------------------------------------------------------------------+
bool ChartDownColorSet(const color clr,const long chart_ID=0)
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur de la barre en bas, de l'ombre et de l'encadrement du co
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_DOWN,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_CHART_LINE** - la couleur de la ligne du graphique et des chandeliers japonais "Doji".

```
//+------------------------------------------------------------------------------------
//| La fonction reçoit la couleur de la ligne du graphique et des chandeliers japonais
//+------------------------------------------------------------------------------------
color ChartLineColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur de la ligne du graphique et des chandeliers japonais "Doji"
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_LINE,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur de la ligne du graphique          |
//| et des chandeliers japonais "Doji".                              |
//+------------------------------------------------------------------+
bool ChartLineColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur de la ligne du graphique et des chandeliers japonais "Do
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_LINE,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

© 2000-2015, MetaQuotes Software Corp.

```
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BULL** - la couleur du corps de la bougie d'haussier.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la couleur du corps de la bougie d'haussier   |
//+------------------------------------------------------------------+
color ChartBullColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur du corps de la bougie d'haussier
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur du corps de la bougie d'haussier  |
//+------------------------------------------------------------------+
bool ChartBullColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur du corps de la bougie d'haussier
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BEAR** -la couleur du corps de la bougie de baissier.

```
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| La fonction reçoit la couleur du corps de la bougie de baissier  |
//+------------------------------------------------------------------+
color ChartBearColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur du corps de la bougie de baissier
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur du corps de la bougie de baissier |
//+------------------------------------------------------------------+
bool ChartBearColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur du corps de la bougie de baissier
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_BID** -la couleur de la ligne du prix Bid.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la couleur de l'affichage de la ligne Bid     |
//+------------------------------------------------------------------+
color ChartBidColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur de la ligne du prix Bid
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BID,0,result))
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       //---  déduisons le message sur l'erreur au journal "Experts"
       Print(__FUNCTION__+", Error Code = ",GetLastError());
      }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur de l'affichage de la ligne Bid    |
//+------------------------------------------------------------------+
bool ChartBidColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur de la ligne du prix Bid
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BID,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_ASK** - la couleur de la ligne du prix Ask.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la couleur de l'affichage de la ligne Ask     |
//+------------------------------------------------------------------+
color ChartAskColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur de la ligne du prix Ask
   if(!ChartGetInteger(chart_ID,CHART_COLOR_ASK,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur de l'affichage de la ligne Ask    |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool ChartAskColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur de la ligne du prix Ask
   if(!ChartSetInteger(chart_ID,CHART_COLOR_ASK,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_LAST** -la couleur de la ligne du prix du dernier marché passé (Last).

```
//+------------------------------------------------------------------------------+
//| La fonction reçoit la couleur de la ligne du prix du dernier marché passé |
//+------------------------------------------------------------------------------+
color ChartLastColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur de la ligne du prix du dernier marché passé (Last)
   if(!ChartGetInteger(chart_ID,CHART_COLOR_LAST,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur de la ligne du prix du dernier    |
//| marché passé.                                                    |
//+------------------------------------------------------------------+
bool ChartLastColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur de la ligne du prix du dernier marché passé (Last)
   if(!ChartSetInteger(chart_ID,CHART_COLOR_LAST,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
```

© 2000-2015, MetaQuotes Software Corp.

```
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_COLOR_STOP_LEVEL** - la couleur des niveaux des ordres stop (Stop Loss et Take Profit).

```
//+------------------------------------------------------------------+
//| La fonction reçoit les couleurs des niveaux Stop Loss et Take Profit |
//+------------------------------------------------------------------+
color ChartStopLevelColorGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir la couleur
   long result=clrNONE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la couleur des niveaux des ordres stop (Stop Loss et Take Profit)
   if(!ChartGetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return((color)result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la couleur des niveaux Stop Loss et Take Profit |
//+------------------------------------------------------------------+
bool ChartStopLevelColorSet(const color clr,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la couleur des niveaux des ordres stop (Stop Loss et Take Profit)
   if(!ChartSetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,clr))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHOW_TRADE_LEVELS** - la propriété de l'affichage des niveaux commerciaux sur le graphique (les niveaux des positions ouvertes, Stop Loss, Take Profit et des ordres remis).

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------------------+
//| La fonction définit, si les niveaux commerciaux sont affichés sur le graphique |
//+------------------------------------------------------------------------------+
bool ChartShowTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------------+
//| La fonction active/désactive le mode de l'affichage des niveaux commerciaux |
//+------------------------------------------------------------------------------+
bool ChartShowTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_DRAG_TRADE_LEVELS** - la propriété de la permission de faire glisser les niveaux commerciaux sur le graphique à l'aide de la souris.

```
//+------------------------------------------------------------------+
//| La fonction définit, si on peut glisser les niveaux commerciaux  |
//| sur le graphique à l'aide de la souris.                          |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsGet(bool &result,const long chart_ID=0)
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive le mode de l'affichage des niveaux  |
//| commerciaux sur le graphique à l'aide de la souris.              |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHOW_DATE_SCALE** - la propriété de l'affichage de l'échelle du temps sur le graphique.

```
//+------------------------------------------------------------------+
//| La fonction définit, si l'échelle du temps sur le graphique est affichée |
//+------------------------------------------------------------------+
bool ChartShowDateScaleGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       //---  déduisons le message sur l'erreur au journal "Experts"
       Print(__FUNCTION__+", Error Code = ",GetLastError());
       return(false);
      }
//--- rappelons la valeur de la propriété du graphique à la variable
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive l'affichage de l'échelle du temps   |
//| sur le graphique.                                                |
//+------------------------------------------------------------------+
bool ChartShowDateScaleSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHOW_PRICE_SCALE** - la propriété de l'affichage de l'échelle du temps sur le graphique.

```
//+------------------------------------------------------------------------+
//| La fonction définit, si l'échelle du temps est affichée sur le graphique |
//+------------------------------------------------------------------------+
bool ChartShowPriceScaleGet(bool &result,const long chart_ID=0)
  {
//--- préparons la variable pour la réception de la valeur de la propriété
   long value;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- rappelons la valeur de la propriété du graphique à la variable
```

© 2000-2015, MetaQuotes Software Corp.

```
   result=value;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction active/désactive le mode de l'affichage de l'échelle |
//| du temps sur le graphique.                                       |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleSet(const bool value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_SHOW_ONE_CLICK** - отображение на графике панели быстрой торговли (опция "Торговля одним кликом").

```
//+------------------------------------------------------------------+
//| Функция определяет, отображается ли на графике                   |
//| панель быстрой торговли ("Торговля одним кликом")                |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelGet(bool &result,const long chart_ID=0)
  {
//--- подготовим переменную для получения значения свойства
   long value;
//--- сбросим значение ошибки
   ResetLastError();
//--- получим значение свойства
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- запомним в переменную значение свойства графика
   result=value;
//--- успешное выполнение
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Функция включает/выключает режим отображения                     |
//| панели быстрой торговли на графике                               |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelSet(const bool value,const long chart_ID=0)
  {
//--- сбросим значение ошибки
   ResetLastError();
//--- установим значение свойства
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- выведем сообщение об ошибке в журнал "Эксперты"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- успешное выполнение
   return(true);
  }
```

- **CHART_SHIFT_SIZE** - la taille de l'alinéa de la barre nulle du bord droit en pourcentage.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la taille de l'alinéa de la barre nulle       |
//| du bord droit du graphique en pourcentage (de 10 % à 50 %).      |
//+------------------------------------------------------------------+
double ChartShiftSizeGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir le résultat
   double result=EMPTY_VALUE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_SHIFT_SIZE,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return(result);
  }
//+------------------------------------------------------------------------+
//| La fonction établit la taille de l'alinéa de la barre nulle du bord    |
//| droit du graphique en pourcentage (de 10 % à 50 %). Pour activer le mode |
//| de l'alinéa il faut établir la valeur de la propriété CHART_SHIFT égal au |
//| true.                                                                  |
//+------------------------------------------------------------------------+
bool ChartShiftSizeSet(const double value,const long chart_ID=0)
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetDouble(chart_ID,CHART_SHIFT_SIZE,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_FIXED_POSITION** - la situation de la position fixée du graphique du bout gauche en pourcentage.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la situation de la position fixée du graphique |
//| du bout gauche en pourcentage.                                   |
//+------------------------------------------------------------------+
double ChartFixedPositionGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir le résultat
   double result=EMPTY_VALUE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_FIXED_POSITION,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return(result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la situation de la position fixée du         |
//| graphique du bout gauche en pourcentage. Pour voir la situation  |
//| de la position fixée sur le graphique il faut préalablement      |
//| spécifier la valeur de la propriété CHART_AUTOSCROLL égal au false |
//+------------------------------------------------------------------+
bool ChartFixedPositionSet(const double value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetDouble(chart_ID,CHART_FIXED_POSITION,value))
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_FIXED_MAX** - la propriété du maximum fixé du graphique.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la valeur du maximum fixé du graphique        |
//+------------------------------------------------------------------+
double ChartFixedMaxGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir le résultat
   double result=EMPTY_VALUE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MAX,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return(result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la valeur du maximum fixé du graphique.      |
//| Pour pouvoir changer la valeur de cette propriété, il faut       |
//| préalablement spécifier la valeur de la propriété                |
//| CHART_SCALEFIX égal au true.                                     |
//+------------------------------------------------------------------+
bool ChartFixedMaxSet(const double value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MAX,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

- **CHART_FIXED_MIN** - la propriété du minimum fixé du graphique.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la valeur du minimum fixé du graphique.       |
//+------------------------------------------------------------------+
double ChartFixedMinGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir le résultat
   double result=EMPTY_VALUE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MIN,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return(result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la valeur du maximum fixé du graphique.      |
//| Pour pouvoir changer la valeur de cette propriété, il faut       |
//| préalablement spécifier la valeur de la propriété CHART_SCALEFIX |
//| égal au true.                                                    |
//+------------------------------------------------------------------+
bool ChartFixedMinSet(const double value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MIN,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_POINTS_PER_BAR** - la valeur de l'échelle en points sur une barre.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la valeur de l'échelle en points sur une barre |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
double ChartPointsPerBarGet(const long chart_ID=0)
  {
//--- préparons la variable pour recevoir le résultat
   double result=EMPTY_VALUE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_POINTS_PER_BAR,0,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return(result);
  }
//+------------------------------------------------------------------+
//| La fonction établit la valeur de l'échelle du graphique en points |
//| sur une barre. Pour regarder le résultat du changement de la      |
//| valeur de cette propriété, il faut préalablement spécifier la     |
//| valeur de la propriété CHART_SCALE_PT_PER_BAR égal au true.       |
//+------------------------------------------------------------------+
bool ChartPointsPerBarSet(const double value,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetDouble(chart_ID,CHART_POINTS_PER_BAR,value))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

- **CHART_PRICE_MIN** - rend la valeur du minimum du graphique.

```
//+------------------------------------------------------------------+
//| La fonction reçoit la valeur du minimum du graphique dans une    |
//| fenêtre principale ou dans un sous - fenêtre.                    |
//+------------------------------------------------------------------+
double ChartPriceMin(const long chart_ID=0,const int sub_window=0)
  {
//--- préparons la variable pour recevoir le résultat
   double result=EMPTY_VALUE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- recevrons la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MIN,sub_window,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return(result);
  }
```

- **CHART_PRICE_MAX** - rend la valeur du maximum du graphique.

```
//+------------------------------------------------------------------+
//| La fonction reçoit rend la valeur du maximum du graphique        |
//| dans une fenêtre principale ou dans un sous - fenêtre.           |
//+------------------------------------------------------------------+
double ChartPriceMax(const long chart_ID=0,const int sub_window=0)
  {
//--- préparons la variable pour recevoir le résultat
   double result=EMPTY_VALUE;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MAX,sub_window,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
     }
//--- rendons la valeur de la propriété du graphique
   return(result);
  }
```

- **CHART_COMMENT** - le texte du commentaire sur le graphique.

```
//+------------------------------------------------------------------------------------------
//| La fonction reçoit le texte du commentaire dans l'angle gauche supérieur du graphi
//+------------------------------------------------------------------------------------------
bool ChartCommentGet(string &result,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons la valeur de la propriété
   if(!ChartGetString(chart_ID,CHART_COMMENT,result))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
```

© 2000-2015, MetaQuotes Software Corp.

```
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction établit le texte du commentaire dans l'angle         |
//| gauche supérieur du graphique.                                   |
//+------------------------------------------------------------------+
bool ChartCommentSet(const string str,const long chart_ID=0)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la valeur de la propriété
   if(!ChartSetString(chart_ID,CHART_COMMENT,str))
     {
      //---  déduisons le message sur l'erreur au journal "Experts"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

Le panneau pour les propriétés du graphique

```
//--- connectons la bibliothèque des éléments de la gestion
#include <ChartObjects\ChartObjectsTxtControls.mqh>
//--- les constantes prédéterminées
#define X_PROPERTY_NAME_1    10  // la coordonnée x du nom de la propriété dans une pr
#define X_PROPERTY_VALUE_1   225 // la coordonnée x de la valeur de la propriété dans
#define X_PROPERTY_NAME_2    345 // la coordonnée x du nom de la propriété dans une de
#define X_PROPERTY_VALUE_2   550 // la coordonnée x de la valeur de la propriété dans
#define X_BUTTON_1           285 // la coordonnée x du bouton dans une première colonn
#define X_BUTTON_2           700 // la coordonnée x du bouton dans une deuxième colonn
#define Y_PROPERTY_1         30  // la coordonnée y du commencement de la première et
#define Y_PROPERTY_2         286 // la coordonnée y du commencement de la troisième co
#define Y_DISTANCE           16  // la distance selon l'axe y entre les lignes
#define LAST_PROPERTY_NUMBER 111 // le numéro de la dernière propriété graphique
//--- les paramètres d'entrée
input color InpFirstColor=clrDodgerBlue; // La couleur des lignes impaires
input color InpSecondColor=clrGoldenrod; // La couleur des lignes paires
//--- les variables et les tableaux
CChartObjectLabel  ExtLabelsName[];  // les inscriptions pour l'affichage des noms des
CChartObjectLabel  ExtLabelsValue[]; // les inscriptions pour l'affichage des valeurs
CChartObjectButton ExtButtons[];     // les boutons
int                ExtNumbers[];     // les indices des propriétés
string             ExtNames[];       // les noms des propriétés
uchar              ExtDataTypes[];   // les types des données des propriétés (integer,
```

© 2000-2015, MetaQuotes Software Corp.

```
uint               ExtGroupTypes[];  // le tableau, qui sauvegarde les données sur l'a
uchar              ExtDrawTypes[];   // le tableau, qui sauvegarde les données sur le
double             ExtMaxValue[];    // les valeurs maximales des propriétés, qu'ils p
double             ExtMinValue[];    // les valeurs minimales des propriétés, qu'ils p
double             ExtStep[];        // les pas pour les changements des propriétés
int                ExtCount;         // le total de toutes les propriétés
color              ExtColors[2];     // le tableau des couleurs pour l'affichage des l
string             ExtComments[2];   // le tableau des commentaires (pour la propriété
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- déduisons le commentaire sur le graphique
   Comment("SomeComment");
//--- sauvegardons les couleurs au tableau pour le basculement ultérieure entre eux
   ExtColors[0]=InpFirstColor;
   ExtColors[1]=InpSecondColor;
//--- sauvegardons les commentaires au tableau pour le basculement ultérieure entre eu
   ExtComments[0]="FirstComment";
   ExtComments[1]="SecondComment";
//--- préparerons et afficheront le panneau pour la gestion des propriétés du graphiqu
   if(!PrepareControls())
      return(INIT_FAILED);
//--- l'exécution réussie
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Deinitialization function of the expert                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- nettoyons le texte du commentaire sur le graphique
   Comment("");
  }
//+------------------------------------------------------------------+
//| Le gestionnaire des événements du graphique                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- la vérification de l'événement de la pression sur l'objet du graphique
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      //--- divisons le nom de l'objet selon le délimiteur
      string obj_name[];
      StringSplit(sparam,'_',obj_name);
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- la vérification, si l'objet est le bouton
      if(obj_name[0]=="Button")
        {
         //--- recevons l'indice du bouton
         int index=(int)StringToInteger(obj_name[1]);
         //--- établissons le bouton à l'état non appuyé
         ExtButtons[index].State(false);
         //--- établissons une nouvelle valeur de la propriété en fonction de son type
         if(ExtDataTypes[index]=='I')
            ChangeIntegerProperty(index);
         if(ExtDataTypes[index]=='D')
            ChangeDoubleProperty(index);
         if(ExtDataTypes[index]=='S')
            ChangeStringProperty(index);
        }
     }
//--- le dessin des valeurs des propriétés
   RedrawProperties();
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Le changement de la propriété entière du graphique               |
//+------------------------------------------------------------------+
void ChangeIntegerProperty(const int index)
  {
//--- recevrons la valeur courante de la propriété
   long value=ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index]);
//--- définissons la valeur suivante de la propriété
   switch(ExtDrawTypes[index])
     {
      case 'C':
         value=GetNextColor((color)value);
         break;
      default:
         value=(long)GetNextValue((double)value,index);
         break;
     }
//--- établissons une nouvelle valeur de la propriété
   ChartSetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index],0,value);
  }
//+------------------------------------------------------------------+
//| Le changement de la propriété matérielle du graphique            |
//+------------------------------------------------------------------+
void ChangeDoubleProperty(const int index)
  {
//--- recevrons la valeur courante de la propriété
   double value=ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index]);
//--- définissons la valeur suivante de la propriété
   value=GetNextValue(value,index);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- établissons une nouvelle valeur de la propriété
   ChartSetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index],value);
  }
//+------------------------------------------------------------------+
//| Le changement de la propriété de chaîne du graphique             |
//+------------------------------------------------------------------+
void ChangeStringProperty(const int index)
  {
//--- la variable statique pour le basculement à l'intérieur du tableau des commentai
   static uint comment_index=1;
//--- changeons l'indice pour la réception de l'autre commentaire
   comment_index=1-comment_index;
//--- établissons une nouvelle valeur de la propriété
   ChartSetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[index],ExtComments[comment_
  }
//+------------------------------------------------------------------+
//| La définition de la valeur suivante de la propriété              |
//+------------------------------------------------------------------+
double GetNextValue(const double value,const int index)
  {
   if(value+ExtStep[index]<=ExtMaxValue[index])
      return(value+ExtStep[index]);
   else
      return(ExtMinValue[index]);
  }
//+------------------------------------------------------------------------+
//| La réception de la couleur suivante pour la propriété du type color |
//+------------------------------------------------------------------------+
color GetNextColor(const color clr)
  {
//---rendons la valeur suivante de la couleur
   switch(clr)
     {
      case clrWhite: return(clrRed);
      case clrRed:   return(clrGreen);
      case clrGreen: return(clrBlue);
      case clrBlue:  return(clrBlack);
      default:       return(clrWhite);
     }
  }
//+------------------------------------------------------------------+
//| Le dessin des valeurs des propriétés                             |
//+------------------------------------------------------------------+
void RedrawProperties(void)
  {
//--- le texte de la valeur de la propriété
   string text;
   long   value;
//--- le cycle par la quantité des propriétés
```

© 2000-2015, MetaQuotes Software Corp.

```
   for(int i=0;i<ExtCount;i++)
     {
      text="";
      switch(ExtDataTypes[i])
        {
         case 'I':
            //--- recevrons la valeur courante de la propriété
            if(!ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[i],0,value))
            break;
            //--- le texte de la propriété entière
            switch(ExtDrawTypes[i])
              {
               //--- la propriété de la couleur
               case 'C':
                  text=(string)((color)value);
                  break;
                  //--- une propriété booléenne
               case 'B':
                  text=(string)((bool)value);
                  break;
                  //--- la propriété de l'enumération ENUM_CHART_MODE
               case 'M':
                  text=EnumToString((ENUM_CHART_MODE)value);
                  break;
                  //--- la propriété de l'enumération ENUM_CHART_VOLUME_MODE
               case 'V':
                  text=EnumToString((ENUM_CHART_VOLUME_MODE)value);
                  break;
                  //--- le nombre du type int
               default:
                  text=IntegerToString(value);
                  break;
              }
            break;
         case 'D':
            //--- le texte de la propriété matérielle
            text=DoubleToString(ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumber
            break;
         case 'S':
            //--- le texte de la propriété de chaîne
            text=ChartGetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[i]);
            break;
        }
      //--- affichons la valeur de la propriété
      ExtLabelsValue[i].Description(text);
     }
  }
//+------------------------------------------------------------------+
//| La création du panneau pour la gestion des propriétés du graphique |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
bool PrepareControls(void)
  {
//--- allouons de la mémoire pour les tableaux avec une réserve
   MemoryAllocation(LAST_PROPERTY_NUMBER+1);
//--- les variables
   int i=0;     // la variable du cycle
   int col_1=0; // le nombre de propriétés dans une première colonne
   int col_2=0; // le nombre de propriétés dans une deuxième colonne
   int col_3=0; // le nombre de propriétés dans une troisième colonne
//--- le nombre actuel de propriétés- 0
   ExtCount=0;
//--- recherchons les propriétés dans le cycle
   while(i<=LAST_PROPERTY_NUMBER)
     {
      //--- rappelons le nombre actuel de propriétés
      ExtNumbers[ExtCount]=i;
      //--- augmentons la valeur de la variable du cycle
      i++;
      //--- vérifions, s'il y a une propriété avec un tel numéro
      if(CheckNumber(ExtNumbers[ExtCount],ExtNames[ExtCount],ExtDataTypes[ExtCount],Ex
        {
         //--- créons les éléments de la gestion pour la propriété
         switch(ExtGroupTypes[ExtCount])
           {
            case 1:
               //--- créons les inscriptions et le bouton pour la propriété
               if(!ShowProperty(ExtCount,0,X_PROPERTY_NAME_1,X_PROPERTY_VALUE_1,X_BUTT
               return(false);
               //--- le nombre d'éléments dans une première colonne a augmenté
               col_1++;
               break;
            case 2:
               //--- créons les inscriptions et le bouton pour la propriété
               if(!ShowProperty(ExtCount,1,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,X_BUTT
               return(false);
               //--- le nombre d'éléments dans une deuxième colonne a augmenté
               col_2++;
               break;
            case 3:
               //--- créons seulement l'inscription pour la propriété
               if(!ShowProperty(ExtCount,2,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,0,Y_PR
               return(false);
               //--- le nombre d'éléments dans une troisième colonne a augmenté
               col_3++;
               break;
           }
         //--- définirons la valeur maximale et minimale de la propriété et le pas
         GetMaxMinStep(ExtNumbers[ExtCount],ExtMaxValue[ExtCount],ExtMinValue[ExtCount
```

© 2000-2015, MetaQuotes Software Corp.

```
         //--- augmentons le nombre de propriétés
         ExtCount++;
        }
     }
//--- libérerons la mémoire, qui n'est pas utilisée par les tableaux
   MemoryAllocation(ExtCount);
//--- redessinons les valeurs des propriétés
   RedrawProperties();
   ChartRedraw();
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| L'allocation de la mémoire pour les tableaux                     |
//+------------------------------------------------------------------+
void MemoryAllocation(const int size)
  {
   ArrayResize(ExtLabelsName,size);
   ArrayResize(ExtLabelsValue,size);
   ArrayResize(ExtButtons,size);
   ArrayResize(ExtNumbers,size);
   ArrayResize(ExtNames,size);
   ArrayResize(ExtDataTypes,size);
   ArrayResize(ExtGroupTypes,size);
   ArrayResize(ExtDrawTypes,size);
   ArrayResize(ExtMaxValue,size);
   ArrayResize(ExtMinValue,size);
   ArrayResize(ExtStep,size);
  }
//+------------------------------------------------------------------+
//| Vérifions l'indice de la propriété sur l'appartenance à un des   |
//| enumérations ENUM_CHART_PROPERTIES                               |
//+------------------------------------------------------------------+
bool CheckNumber(const int ind,string &name,uchar &data_type,uint &group_type,uchar &d
  {
//--- vérifions si la propriété est entière
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_INTEGER)ind);
   if(_LastError==0)
     {
      data_type='I';                        // la propriété de l'enumération ENUM_CHART_
      GetTypes(ind,group_type,draw_type); // définirons les paramètres de l'affichage
      return(true);
     }
//--- vérifions si la propriété est matérielle
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_DOUBLE)ind);
   if(_LastError==0)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      data_type='D';                        // la propriété de l'enumération ENUM_CHART_
      GetTypes(ind,group_type,draw_type); // définirons les paramètres de l'affichage
      return(true);
     }
//--- vérifions si la propriété est de chaîne
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_STRING)ind);
   if(_LastError==0)
     {
      data_type='S';                        // la propriété de l'enumération ENUM_CHART_
      GetTypes(ind,group_type,draw_type); // définirons les paramètres de l'affichage
      return(true);
     }
//--- La propriété n'appartient pas à aucune enumération
   return(false);
  }
//+------------------------------------------------------------------------------+
//| La définition de celui-là, dans quel groupe doit se trouver la propriété,  |
//| et le type de l'affichage                                                    |
//+------------------------------------------------------------------------------+
void GetTypes(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- vérifions sur l'appartenance de la propriété au troisième groupe
//--- les propriétés du troisième groupe s'affichent dans une deuxième colonne, à part
   if(CheckThirdGroup(property_number,group_type,draw_type))
      return;
//--- vérifions sur l'appartenance de la propriété au deuxième groupe
//--- les propriétés du deuxième groupe s'affichent dans une deuxième colonne, à parti
   if(CheckSecondGroup(property_number,group_type,draw_type))
      return;
//--- si vous êtes ici, cela signifie que la propriété appartient au premier groupe (1
   CheckFirstGroup(property_number,group_type,draw_type);
  }
//+------------------------------------------------------------------+
//| La fonction vérifie, si la propriété appartient au troisième     |
//| groupe, et si oui, cela définit son type de l'affichage          |
//+------------------------------------------------------------------+
bool CheckThirdGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- vérifions la propriété sur l'appartenance au troisième groupe
   switch(property_number)
     {
      //--- les propriétés booléennes
      case CHART_IS_OBJECT:
      case CHART_WINDOW_IS_VISIBLE:
         draw_type='B';
         break;
         //--- les propriétés entières
      case CHART_VISIBLE_BARS:
```

© 2000-2015, MetaQuotes Software Corp.

```
      case CHART_WINDOWS_TOTAL:
      case CHART_WINDOW_HANDLE:
      case CHART_WINDOW_YDISTANCE:
      case CHART_FIRST_VISIBLE_BAR:
      case CHART_WIDTH_IN_BARS:
      case CHART_WIDTH_IN_PIXELS:
         draw_type='I';
         break;
         //--- les propriétés matérielles
      case CHART_PRICE_MIN:
      case CHART_PRICE_MAX:
         draw_type='D';
         break;
         //--- en fait, cette propriété est la commande de l'affichage du graphique au
         //--- il n'y a pas de nécessité pour ce panneau pour son application, puisque
         //--- soit toujours par-dessus tout les autres plus tôt, que nous l'utilisons
      case CHART_BRING_TO_TOP:
         draw_type=' ';
         break;
         //--- la propriété n'appartient pas au troisième groupe
      default:
         return(false);
     }
//--- la propriété appartient au troisième groupe
   group_type=3;
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction vérifie, si la propriété appartient au deuxième groupe, et |
//| si oui, cela définit son type de l'affichage                     |
//+------------------------------------------------------------------+
bool CheckSecondGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- vérifions la propriété sur l'appartenance au deuxième groupe
   switch(property_number)
     {
      //--- la propriété du type ENUM_CHART_MODE
      case CHART_MODE:
         draw_type='M';
         break;
         //--- la propriété du type ENUM_CHART_VOLUME_MODE
      case CHART_SHOW_VOLUMES:
         draw_type='V';
         break;
         //--- la propriété de chaîne
      case CHART_COMMENT:
         draw_type='S';
         break;
         //--- la propriété de la couleur
```

© 2000-2015, MetaQuotes Software Corp.

```
      case CHART_COLOR_BACKGROUND:
      case CHART_COLOR_FOREGROUND:
      case CHART_COLOR_GRID:
      case CHART_COLOR_VOLUME:
      case CHART_COLOR_CHART_UP:
      case CHART_COLOR_CHART_DOWN:
      case CHART_COLOR_CHART_LINE:
      case CHART_COLOR_CANDLE_BULL:
      case CHART_COLOR_CANDLE_BEAR:
      case CHART_COLOR_BID:
      case CHART_COLOR_ASK:
      case CHART_COLOR_LAST:
      case CHART_COLOR_STOP_LEVEL:
         draw_type='C';
         break;
         //--- la propriété n'appartient pas au deuxième groupe
      default:
         return(false);
     }
//--- la propriété appartient au deuxième groupe
   group_type=2;
   return(true);
  }
//+------------------------------------------------------------------+
//| Cette fonction est appelée seulement dans le cas où on sait déjà,  |
//| que la propriété n'appartient pas aux deuxième et troisième groupes des propriétés
//+------------------------------------------------------------------+
void CheckFirstGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- la propriété appartient au premier groupe
   group_type=1;
//--- définirons le type de l'affichage de la propriété
   switch(property_number)
     {
      //--- les propriétés entières
      case CHART_SCALE:
      case CHART_HEIGHT_IN_PIXELS:
         draw_type='I';
         return;
         //--- les propriétés matérielles
      case CHART_SHIFT_SIZE:
      case CHART_FIXED_POSITION:
      case CHART_FIXED_MAX:
      case CHART_FIXED_MIN:
      case CHART_POINTS_PER_BAR:
         draw_type='D';
         return;
         //---sont restés seulement les propriétés booléennes
      default:
```

© 2000-2015, MetaQuotes Software Corp.

```
         draw_type='B';
         return;
      }
   }
//+------------------------------------------------------------------+
//| La création de l'inscription et du bouton pour la propriété
//+------------------------------------------------------------------+
bool ShowProperty(const int ind,const int type,const int x1,const int x2,
                  const int xb,const int y,const bool btn)
   {
//--- le tableau statique pour le basculement à l'intérieur du tableau de la couleur E
   static uint color_index[3]={1,1,1};
//--- changeons l'indice pour la réception de l'autre couleur
   color_index[type]=1-color_index[type];
//--- déduisons les inscriptions et le bouton (si btn=true) pour la propriété
   if(!LabelCreate(ExtLabelsName[ind],"name_"+(string)ind,ExtNames[ind],ExtColors[colo
      return(false);
   if(!LabelCreate(ExtLabelsValue[ind],"value_"+(string)ind,"",ExtColors[color_index[t
      return(false);
   if(btn && !ButtonCreate(ExtButtons[ind],(string)ind,xb,y+1))
      return(false);
//--- l'exécution réussie
   return(true);
   }
//+------------------------------------------------------------------+
//| La création de l'inscription                                     |
//+------------------------------------------------------------------+
bool LabelCreate(CChartObjectLabel &lbl,const string name,const string text,
                 const color clr,const int x,const int y)
   {
   if(!lbl.Create(0,"Label_"+name,0,x,y)) return(false);
   if(!lbl.Description(text))             return(false);
   if(!lbl.FontSize(10))                  return(false);
   if(!lbl.Color(clr))                    return(false);
//--- l'exécution réussie
   return(true);
   }
//+------------------------------------------------------------------+
//| La création du bouton                                            |
//+------------------------------------------------------------------+
bool ButtonCreate(CChartObjectButton &btn,const string name,
                  const int x,const int y)
   {
   if(!btn.Create(0,"Button_"+name,0,x,y,50,15)) return(false);
   if(!btn.Description("Next"))                  return(false);
   if(!btn.FontSize(10))                         return(false);
   if(!btn.Color(clrBlack))                      return(false);
   if(!btn.BackColor(clrWhite))                  return(false);
   if(!btn.BorderColor(clrBlack))                return(false);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Définirons la valeur maximale et minimale de la propriété et le pas  |
//+------------------------------------------------------------------+
void GetMaxMinStep(const int property_number,double &max,double &min,double &step)
  {
   double value;
//--- établissons les valeurs en fonction du type de la propriété
   switch(property_number)
     {
      case CHART_SCALE:
         max=5;
         min=0;
         step=1;
         break;
      case CHART_MODE:
      case CHART_SHOW_VOLUMES:
         max=2;
         min=0;
         step=1;
         break;
      case CHART_SHIFT_SIZE:
         max=50;
         min=10;
         step=2.5;
         break;
      case CHART_FIXED_POSITION:
         max=90;
         min=0;
         step=15;
         break;
      case CHART_POINTS_PER_BAR:
         max=19;
         min=1;
         step=3;
         break;
      case CHART_FIXED_MAX:
         value=ChartGetDouble(0,CHART_FIXED_MAX);
         max=value*1.25;
         min=value;
         step=value/32;
         break;
      case CHART_FIXED_MIN:
         value=ChartGetDouble(0,CHART_FIXED_MIN);
         max=value;
         min=value*0.75;
         step=value/32;
```

© 2000-2015, MetaQuotes Software Corp.

```
         break;
      case CHART_HEIGHT_IN_PIXELS:
         max=700;
         min=520;
         step=30;
         break;
         //--- les valeurs par défaut
      default:
         max=1;
         min=0;
         step=1;
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Les constantes des objets

On prévoit 44 objets graphiques, que l'on peut créer et afficher sur le graphique de prix. Toutes les constantes pour le travail avec les objets sont divisées à 9 groupes:

- Les types des objets – les identificateurs des objets graphiques;
- Les propriétés des objets – le travail avec les propriétés des objets graphiques;
- Les moyens du rattachement des objets – les constantes du positionnement des objets sur le graphique;
- L'angle du rattachement – permet d'établir l'angle du graphique, par rapport à laquel on produit le positionnement de l'objet en pixels;
- La visibilité des objets – spécifier les temps trames, où l'objet est visible;
- Les niveaux des ondes d'Elliott – la gradation du marquage d'onde;
- Les objets de Gann – les constantes de la tendance pour le fanion de Gann et la grille de Gann;
- L'ensemble des couleurs Web – les constantes des couleurs Web prédéterminées;
- Wingdings – les codes des caractères de la fonte Wingdings.

© 2000-2015, MetaQuotes Software Corp.

Les types des objets

A la création de l'objet graphique par la fonction ObjectCreate() il est nécessaire d'indiquer le type de l'objet créé, qui peut accepter une des valeur d'énumération ENUM_OBJECT. Les précisions suivantes des propriétés de l'objet créé sont possibles à l'aide des fonctions du travail avec les objets graphiques.

ENUM_OBJECT

Identificateur		Description
OBJ_VLINE		La ligne verticale
OBJ_HLINE		Ligne horizontale
OBJ_TREND		La ligne de Tendance
OBJ_TRENDBYANGLE		La ligne de tendance par l'angle
OBJ_CYCLES		Les lignes cycliques
OBJ_ARROWED_LINE		L'objet "Line avec une flèche"
OBJ_CHANNEL		Le canal équidistant
OBJ_STDDEVCHANNEL		Le canal de l' écart type
OBJ_REGRESSION		Le canale de la régression linéaire
OBJ_PITCHFORK		La fourche d'Andrews
OBJ_GANNLINE		La ligne de Gann
OBJ_GANNFAN		Le fanion de Gann
OBJ_GANNGRID		La grille de Gann
OBJ_FIBO		Les niveaux de Fibonacci
OBJ_FIBOTIMES		Les zones temporaires de Fibonacci
OBJ_FIBOFAN		Le fanion de Fibonacci
OBJ_FIBOARC		Les arcs de Fibonacci
OBJ_FIBOCHANNEL		Le canale de Fibonacci
OBJ_EXPANSION		L'expansion de Fibonacci
OBJ_ELLIOTWAVE5		Vague d'Elliott pulsatoire
OBJ_ELLIOTWAVE3		Vague d'Elliott corrective
OBJ_RECTANGLE		Le rectangle
OBJ_TRIANGLE		Le triangle
OBJ_ELLIPSE		L'ellipse

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW_THUMB_UP		Le signe "bien" (le pouce en haut)
OBJ_ARROW_THUMB_DOWN		Le signe "mal" (le pouce en bas)
OBJ_ARROW_UP		Le signe "la flèche en haut"
OBJ_ARROW_DOWN		Le signe "la flèche en bas"
OBJ_ARROW_STOP		Le signe "Stop"
OBJ_ARROW_CHECK		Le signe "Vérification"
OBJ_ARROW_LEFT_PRICE		La balise gauche de prix
OBJ_ARROW_RIGHT_PRICE		La balise droite de prix
OBJ_ARROW_BUY		Le signe "Buy"
OBJ_ARROW_SELL		Le signe "Sell"
OBJ_ARROW		L'objet "La flèche"
OBJ_TEXT		L'objet "Le texte"
OBJ_LABEL		L'objet "La balise de texte"
OBJ_BUTTON		L'objet "Le bouton"
OBJ_CHART		L'objet "Le graphique"
OBJ_BITMAP		L'objet "Le dessin"
OBJ_BITMAP_LABEL		L'objet "La balise graphique"
OBJ_EDIT		L'objet "Le champ de l'entrée"
OBJ_EVENT		L'objet "Événement", correspondant à l'événement dans le calendrier économique
OBJ_RECTANGLE_LABEL		L'objet "Marque rectangulaire" pour la création et la présentation de l'interface d'utilisateur graphique.

© 2000-2015, MetaQuotes Software Corp.

OBJ_VLINE

La ligne verticale

Note

A la création de la ligne verticale, on peut indiquer le mode de l'affichage de la ligne à toutes les fenêtres du graphique (la propriété OBJPROP_RAY).

L'exemple

Le script suivant crée et déplace la ligne verticale sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"La ligne verticale\"."
#property description "La date du point du rattachement est spécifiée en pourcentage d
#property description "de la fenêtre du graphique dans les barres."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string           InpName="VLine";     // Le nom de la ligne
input int              InpDate=25;          // La date de la ligne au %
input color            InpColor=clrRed;     // La couleur de la ligne
input ENUM_LINE_STYLE  InpStyle=STYLE_DASH; // Le style de la ligne
input int              InpWidth=3;          // L'épaisseur de la ligne
input bool             InpBack=false;       // La ligne à l'arrière-plan
input bool             InpSelection=true;   // Sélectionner pour les déplacements
input bool             InpRay=true;         // La suite de la ligne en bas
```

© 2000-2015, MetaQuotes Software Corp.

```
input bool            InpHidden=true;      // Est caché dans la liste des objets
input long            InpZOrder=0;         // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée la ligne verticale                                          |
//+------------------------------------------------------------------+
bool VLineCreate(const long            chart_ID=0,        // ID du graphique
                 const string          name="VLine",      // le nom de la ligne
                 const int             sub_window=0,      // le numéro du sous-fenêtre
                 datetime              time=0,            // le temps de la ligne
                 const color           clr=clrRed,        // la couleur de la ligne
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la ligne
                 const int             width=1,           // l'épaisseur de la ligne
                 const bool            back=false,        // à l'arrière-plan
                 const bool            selection=true,    // sélectionner pour les dép
                 const bool            ray=true,          // la suite de la ligne en b
                 const bool            hidden=true,       // est caché dans la liste d
                 const long            z_order=0)         // la priorité au clic d'une
  {
//---si le temps de la ligne n'est pas spécifié, passons-la par la dernière barre
   if(!time)
      time=TimeCurrent();
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons la ligne verticale
   if(!ObjectCreate(chart_ID,name,OBJ_VLINE,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer la ligne verticale! Le code de l'erreur = ",G
      return(false);
     }
//--- établissons la couleur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de l'affichage de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement des lignes par la
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activons (true) ou désactivons (false) le mode de l'affichage de la ligne dans l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY,ray);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Le déplacement de la ligne verticale                             |
//+------------------------------------------------------------------+
bool VLineMove(const long   chart_ID=0,   // ID du graphique
               const string name="VLine", // le nom de la ligne
               datetime     time=0)       // le temps de la ligne
  {
//---si le temps de la ligne n'est pas spécifié, déplaçons celle-ci sur la dernière ba
   if(!time)
      time=TimeCurrent();
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons la ligne verticale
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la ligne verticale! Le code de l'erreur =
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime la ligne verticale                                      |
//+------------------------------------------------------------------+
bool VLineDelete(const long   chart_ID=0,   // ID du graphique
                 const string name="VLine") // le nom de la ligne
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la ligne verticale
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer la ligne verticale! Le code de l'erreur =
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate<0 || InpDate>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- le tableau pour stocker les valeurs des dates qui seront utilisées
//---pour l'établissement et le changement de la coordonnée du point du rattachement d
   datetime date[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- définissons les points pour le dessin de la ligne
   int d=InpDate*(bars-1)/100;
//--- créons la ligne verticale
   if(!VLineCreate(0,InpName,0,date[d],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpRay,InpHidden,InpZOrder))
      return;
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//---maintenant déplaçons la ligne
//--- le compteur du cycle
   int h_steps=bars/2;
//--- déplaçons la ligne
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- deplaçons le point
      if(!VLineMove(0,InpName,date[d]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.03 seconde
      Sleep(30);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le canal du graphique
   VLineDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_HLINE

La ligne horizontale.

L'exemple

Le script suivant crée et déplace la ligne horizontale sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"La ligne horizontale\"."
#property description "Le prix du point du rattachement est spécifié en pourcentage de"
#property description "de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="HLine";      // Le nom de la ligne
input int             InpPrice=25;          // Le prix de la ligne en %
input color           InpColor=clrRed;      // La couleur de la ligne
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Le style de la ligne
input int             InpWidth=3;           // L'épaisseur de la ligne
input bool            InpBack=false;        // La ligne à l'arrière-plan
input bool            InpSelection=true;    // Sélectionner pour les déplacements
input bool            InpHidden=true;       // Est caché dans la liste des objets
input long            InpZOrder=0;          // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée la ligne horizontale                                        |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool HLineCreate(const long            chart_ID=0,        // ID du graphique
                 const string          name="HLine",      // le nom de la ligne
                 const int             sub_window=0,      // le numéro du sous-fenêtre
                 double                price=0,           // le prix de la ligne
                 const color           clr=clrRed,        // la couleur de la ligne
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la ligne
                 const int             width=1,           // l'épaisseur de la ligne
                 const bool            back=false,        // à l'arrière-plan
                 const bool            selection=true,    // sélectionner pour les dép
                 const bool            hidden=true,       // est caché dans la liste d
                 const long            z_order=0)         // la priorité au clic d'une
  {
//--- si le prix n'est pas spécifié, fixerons-la au niveau du prix courant Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons la ligne horizontale
   if(!ObjectCreate(chart_ID,name,OBJ_HLINE,sub_window,0,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer la ligne horizontale! Le code de l'erreur = "
      return(false);
     }
//--- établissons la couleur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de l'affichage de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement des lignes par la
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Le déplacement de la ligne horizontale                           |
//+------------------------------------------------------------------+
bool HLineMove(const long   chart_ID=0,   // ID du graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
                const string name="HLine", // le nom de la ligne
                double       price=0)      // le prix de la ligne
  {
//--- si le prix de la ligne n'est pas spécifié, déplaçons-la au niveau du prix courar
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons la ligne horizontale
   if(!ObjectMove(chart_ID,name,0,0,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la ligne horizontale! Le code de l'erreur
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime la ligne horizontale                                    |
//+------------------------------------------------------------------+
bool HLineDelete(const long   chart_ID=0,   // ID du graphique
                 const string name="HLine") // le nom de la ligne
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la ligne horizontale
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer la ligne horizontale! Le code de l'erreu
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpPrice<0 || InpPrice>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- la taille du tableau price
   int accuracy=1000;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- le tableau pour stocker les valeurs des prix qui seront utilisées
//---pour l'établissement et le changement de la coordonnée du point du rattachement d
   double price[];
//--- l'allocation de la mémoire
   ArrayResize(price,accuracy);
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de la ligne
   int p=InpPrice*(accuracy-1)/100;
//--- créons la ligne horizontale
   if(!HLineCreate(0,InpName,0,price[p],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//---maintenant déplaçons la ligne
//--- le compteur du cycle
   int v_steps=accuracy/2;
//--- déplaçons la ligne
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p<accuracy-1)
         p+=1;
      //--- deplaçons le point
      if(!HLineMove(0,InpName,price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons du graphique
   HLineDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
```

© 2000-2015, MetaQuotes Software Corp.

```
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_TREND

La ligne de tendance.

Note

Pour la ligne de tendance, on peut spécifier le mode de son affichage à droite et/ou à gauche (les propriétésOBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT en conséquence).

L'exemple

Le script suivant crée et déplace la ligne de tendance sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"La ligne de tendance\"
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="Trend";     // Le nom de la ligne
input int             InpDate1=35;         // La date du 1-ier point en %
input int             InpPrice1=60;        // Le prix du 1-ier point en %
input int             InpDate2=65;         // La date du 2-ième point en %
input int             InpPrice2=40;        // Le prix du 2-ième point en %
input color           InpColor=clrRed;     // La couleur de la ligne
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Le style de la ligne
```

© 2000-2015, MetaQuotes Software Corp.

```
input int              InpWidth=2;          // L'épaisseur de la ligne
input bool             InpBack=false;       // La ligne à l'arrière-plan
input bool             InpSelection=true;   // Sélectionner pour les déplacements
input bool             InpRayLeft=false;    // La suite de la ligne à gauche
input bool             InpRayRight=false;   // La suite de la ligne à droite
input bool             InpHidden=true;      // Est caché dans la liste des objets
input long             InpZOrder=0;         // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée la ligne de la tendance selon les coordonnées spécifiées    |
//+------------------------------------------------------------------+
bool TrendCreate(const long            chart_ID=0,        // ID du graphique
                 const string          name="TrendLine",  // le nom de la ligne
                 const int             sub_window=0,      // le numéro du sous-fenêtre
                 datetime              time1=0,           // le temps du premier point
                 double                price1=0,          // le prix du premier point
                 datetime              time2=0,           // le temps du deuxième poi
                 double                price2=0,          // le prix du deuxième point
                 const color           clr=clrRed,        // la couleur de la ligne
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la ligne
                 const int             width=1,           // l'épaisseur de la ligne
                 const bool            back=false,        // à l'arrière-plan
                 const bool            selection=true,    // sélectionner pour les dép
                 const bool            ray_left=false,    // la suite de la ligne à ga
                 const bool            ray_right=false,   // la suite de la ligne à dr
                 const bool            hidden=true,       // est caché dans la liste d
                 const long            z_order=0)         // la priorité au clic d'une
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeTrendEmptyPoints(time1,price1,time2,price2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons la ligne de tendance selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_TREND,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer la ligne de tendance! Le code de l'erreur = "
      return(false);
     }
//--- établissons la couleur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de l'affichage de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement des lignes par la
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage de la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//---activons (true) ou désactivons (false) le mode de la suite de l'affichage de la l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement de la ligne de la tendance
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // ID du graphique
                      const string name="TrendLine", // le nom de la ligne
                      const int    point_index=0,    // le numéro du point du rattache
                      datetime     time=0,           // la coordonnée du temps du poin
                      double       price=0)          // la coordonnée du prix du point
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement de la ligne de la tendance
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction supprime la ligne de la tendance du graphique.
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // ID du graphique
                 const string name="TrendLine") // le nom de la ligne
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la ligne de tendance
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer la ligne de la tendance! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement de la ligne de la tendance, et pour
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- si le temps du premier point n'est pas spécifié, il sera sur la barre courante
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du deuxième point n'est pas spécifié, il se trouve au 9 barres de ga
   if(!time2)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- établissons le deuxième point au 9 barres de gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas spécifié, il coïncide avec le prix du pre
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de la ligne
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- créons la ligne de tendance
   if(!TrendCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,InpStyle
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement de la ligne
//--- le compteur du cycle
   int v_steps=accuracy/5;
//---déplaçons le premier point du rattachement selon la verticale
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1>1)
         p1-=1;
      //--- deplaçons le point
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
```

© 2000-2015, MetaQuotes Software Corp.

```
         return;
       //--- vérifions le fait de l'arrêt forcé du script
       if(IsStopped())
         return;
       //---redessinons le graphique
       ChartRedraw();
      }
//--- déplaçons le deuxième point du rattachement selon la verticale
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p2<accuracy-1)
         p2+=1;
      //--- deplaçons le point
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à une demi-seconde
   Sleep(500);
//--- le compteur du cycle
   int h_steps=bars/2;
//--- déplaçons les deux points du rattachement à l'horizontale simultanément
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons les valeurs suivantes
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- deplaçons les points
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.03 seconde
      Sleep(30);
     }
//--- le retard à 1 seconde
   Sleep(1000);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- supprimons la ligne de tendance
   TrendDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_TRENDBYANGLE

La ligne de tendance selon l'angle.

Note

Pour la ligne de tendance, on peut spécifier le mode de son affichage à droite et/ou à gauche (les propriétésOBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT en conséquence).

Pour l'établissement de l'inclinaison de la ligne on peut utiliser l'angle, ainsi que les coordonnées du deuxième point du rattachement.

L'exemple

Le script suivant crée et déplace la ligne de tendance sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"La ligne de tendance se
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="Trend";      // Le nom de la ligne
input int             InpDate1=50;          // La date du 1-ier point en %
input int             InpPrice1=75;         // Le prix du 1-ier point en %
input int             InpAngle=0;           // L'angle de l'inclinaison de la ligne
input color           InpColor=clrRed;      // La couleur de la ligne
```

© 2000-2015, MetaQuotes Software Corp.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Le style de la ligne
input int             InpWidth=2;          // L'épaisseur de la ligne
input bool            InpBack=false;       // La ligne à l'arrière-plan
input bool            InpSelection=true;   // Sélectionner pour les déplacements
input bool            InpRayLeft=false;    // La suite de la ligne à gauche
input bool            InpRayRight=true;    // La suite de la ligne à droite
input bool            InpHidden=true;      // Est caché dans la liste des objets
input long            InpZOrder=0;         // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée la ligne de la tendance selon l'angle                       |
//+------------------------------------------------------------------+
bool TrendByAngleCreate(const long            chart_ID=0,        // ID du graphique
                        const string          name="TrendLine",  // le nom de la ligne
                        const int             sub_window=0,      // le numéro du sous-
                        datetime              time=0,            // le temps du point
                        double                price=0,           // le prix du point
                        const double          angle=45.0,        // l'angle de l'incli
                        const color           clr=clrRed,        // la couleur de la l
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la lig
                        const int             width=1,           // l'épaisseur de la
                        const bool            back=false,        // à l'arrière-plan
                        const bool            selection=true,    // sélectionner pour
                        const bool            ray_left=false,    // la suite de la lig
                        const bool            ray_right=true,    // la suite de la lig
                        const bool            hidden=true,       // est caché dans la
                        const long            z_order=0)         // la priorité au cli
  {
//--- pour qu'il était confortable de déplacer la ligne de tendance par la souris, cré
   datetime time2=0;
   double   price2=0;
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeTrendEmptyPoints(time,price,time2,price2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- construisons la ligne de tendance selon deux points
   if(!ObjectCreate(chart_ID,name,OBJ_TRENDBYANGLE,sub_window,time,price,time2,price2)
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer la ligne de tendance! Le code de l'erreur = "
      return(false);
     }
//---changeons l'angle de l'inclinaison de la ligne de tendance; en train du changemer
//--- du point de la ligne sera redéfinie automatiquement conformément à une nouvelle
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- établissons la couleur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur de la ligne
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement des lignes par la
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage de la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//---activons (true) ou désactivons (false) le mode de la suite de l'affichage de la l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change les coordonnées du point du rattachement de la ligne de la tendance
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // ID du graphique
                      const string name="TrendLine", // le nom de la ligne
                      datetime     time=0,           // la coordonnée du temps du poi
                      double       price=0)          // la coordonnée du prix du point
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement de la ligne de la tendance
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'er
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change l'angle de l'inclinaison de la ligne de la tendance
//+------------------------------------------------------------------+
bool TrendAngleChange(const long   chart_ID=0,       // ID du graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
                     const string name="TrendLine", // le nom de la ligne de la tenda
                     const double angle=45)         // l'angle de l'inclinaison de la
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons l'angle de l'inclinaison de la ligne de la tendance
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer l'angle de l'inclinaison de la ligne! Le co
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime la ligne de la tendance                                 |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // ID du graphique
                 const string name="TrendLine") // le nom de la ligne
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la ligne de tendance
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer la ligne de la tendance! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement de la ligne de la tendance, et pour
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- si le temps du premier point n'est pas spécifié, il sera sur la barre courante
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- établissons les coordonnées du deuxième point auxiliaire
//--- le deuxième point sera à gauche sur 9 barres et aura le même prix
   datetime second_point_time[10];
```

© 2000-2015, MetaQuotes Software Corp.

```
   CopyTime(Symbol(),Period(),time1,10,second_point_time);
   time2=second_point_time[0];
   price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de la ligne
   int d1=InpDate1*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- créons la ligne de tendance
   if(!TrendByAngleCreate(0,InpName,0,date[d1],price[p1],InpAngle,InpColor,InpStyle,
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons et tournons la ligne
//--- le compteur du cycle
   int v_steps=accuracy/2;
//--- déplaçons le point du rattachement et changeons l'angle de l'inclinaison de la l
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1>1)
         p1-=1;
      //--- deplaçons le point
      if(!TrendPointChange(0,InpName,date[d1],price[p1]))
         return;
      if(!TrendAngleChange(0,InpName,18*(i+1)))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons du graphique
   TrendDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_CYCLES

Lignes cycliques.

Note

La distance entre les lignes est spécifiée aux coordonnées du temps de deux points du rattachement de l'objet.

L'exemple

Le script suivant crée et déplace les lignes cycliques sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit les lignes cycliques sur le graphique."
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "de la dimension de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="Cycles";   // Le nom de l'objet
input int             InpDate1=10;        // La date du 1-ier point en %
input int             InpPrice1=45;       // Le prix du 1-ier point en %
input int             InpDate2=20;        // La date du 2-ième point en %
input int             InpPrice2=55;       // Le prix du 2-ième point en %
input color           InpColor=clrRed;    // La couleur des lignes cycliques
input ENUM_LINE_STYLE InpStyle=STYLE_DOT; // Le style des lignes cycliques
```

© 2000-2015, MetaQuotes Software Corp.

```
input int              InpWidth=1;         // L'épaisseur des lignes cycliques
input bool             InpBack=false;      // L'objet à l'arrière-plan
input bool             InpSelection=true;  // Sélectionner pour les déplacements
input bool             InpHidden=true;     // Est caché dans la liste des objets
input long             InpZOrder=0;        // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée les lignes cycliques                                        |
//+------------------------------------------------------------------+
bool CyclesCreate(const long            chart_ID=0,        // ID du graphique
                  const string          name="Cycles",     // le nom de l'objet
                  const int             sub_window=0,      // le numéro du sous-fenêtr
                  datetime              time1=0,           // le temps du premier poir
                  double                price1=0,          // le prix du premier point
                  datetime              time2=0,           // le temps du deuxième poi
                  double                price2=0,          // le prix du deuxième poir
                  const color           clr=clrRed,        // la couleur des lignes cy
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // le style des lignes cycl
                  const int             width=1,           // l'épaisseur des lignes c
                  const bool            back=false,        // à l'arrière-plan
                  const bool            selection=true,    // sélectionner pour les dé
                  const bool            hidden=true,       // est caché dans la liste
                  const long            z_order=0)         // la priorité au clic d'ur
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeCyclesEmptyPoints(time1,price1,time2,price2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons les lignes cycliques selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_CYCLES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer les lignes cycliques! Le code de l'erreur = "
      return(false);
     }
//--- établissons la couleur des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de l'affichage des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement des lignes par la
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on r
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
bool CyclesPointChange(const long   chart_ID=0,    // ID du graphique
                       const string name="Cycles", // le nom de l'objet
                       const int    point_index=0, // le numéro du point du rattacheme
                       datetime     time=0,        // la coordonnée du temps du point
                       double       price=0)       // la coordonnée du prix du point d
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime les lignes cycliques                                    |
//+------------------------------------------------------------------+
bool CyclesDelete(const long   chart_ID=0,    // ID du graphique
                  const string name="Cycles") // le nom de l'objet
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons les lignes cycliques
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer les lignes cycliques! Le code de l'erreur
      return(false);
     }
//--- l'exécution réussie
   return(true);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement des lignes cycliques, et pour les v
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeCyclesEmptyPoints(datetime &time1,double &price1,
                             datetime &time2,double &price2)
  {
//--- si le temps du premier point n'est pas spécifié, il sera sur la barre courante
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du deuxième point n'est pas spécifié, il se trouve au 9 barres de ga
   if(!time2)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- établissons le deuxième point au 9 barres de gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas spécifié, il coïncide avec le prix du pre
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
```

© 2000-2015, MetaQuotes Software Corp.

```
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin des lignes cycliques
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- créons la ligne de tendance
   if(!CyclesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement
//--- le compteur du cycle
   int h_steps=bars/5;
//--- déplaçons le deuxième point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d2<bars-1)
         d2+=1;
      //--- deplaçons le point
      if(!CyclesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   h_steps=bars/4;
//--- déplaçons le premier point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d1<bars-1)
         d1+=1;
      //--- deplaçons le point
      if(!CyclesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons l'objet du graphique
   CyclesDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROWED_LINE

La ligne avec la flèche.

L'exemple

Le script suivant crée et déplace la ligne avec la flèche sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"La ligne avec la flèche
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="ArrowedLine"; // Le nom de la ligne
input int             InpDate1=35;           // La date du 1-ier point en %
input int             InpPrice1=60;          // Le prix du 1-ier point en %
input int             InpDate2=65;           // La date du 2-ième point en %
input int             InpPrice2=40;          // Le prix du 2-ième point en %
input color           InpColor=clrRed;       // La couleur de la ligne
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;   // Le style de la ligne
input int             InpWidth=2;            // L'épaisseur de la ligne
input bool            InpBack=false;         // La ligne à l'arrière-plan
input bool            InpSelection=true;     // Sélectionner pour les déplacements
input bool            InpHidden=true;        // Est caché dans la liste des objets
input long            InpZOrder=0;           // La priorité au clic d'une souris
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Crée la ligne avec la flèche selon les coordonnées spécifiées    |
//+------------------------------------------------------------------+
bool ArrowedLineCreate(const long            chart_ID=0,         // ID du graphique
                       const string          name="ArrowedLine", // le nom de la ligne
                       const int             sub_window=0,       // le numéro du sous-
                       datetime              time1=0,            // le temps du premie
                       double                price1=0,           // le prix du premier
                       datetime              time2=0,            // le temps du deuxié
                       double                price2=0,           // le prix du deuxièm
                       const color           clr=clrRed,         // la couleur de la l
                       const ENUM_LINE_STYLE style=STYLE_SOLID,  // le style de la lig
                       const int             width=1,            // l'épaisseur de la
                       const bool            back=false,         // à l'arrière-plan
                       const bool            selection=true,     // sélectionner pour
                       const bool            hidden=true,        // est caché dans la
                       const long            z_order=0)          // la priorité au cli
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeArrowedLineEmptyPoints(time1,price1,time2,price2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons la ligne avec la flèche selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_ARROWED_LINE,sub_window,time1,price1,time2,price
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer la ligne avec la flèche! Le code de l'erreur
      return(false);
     }
//--- établissons la couleur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de l'affichage de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement des lignes par la
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement de la ligne avec la flèche      |
//+------------------------------------------------------------------+
bool ArrowedLinePointChange(const long   chart_ID=0,          // ID du graphique
                            const string name="ArrowedLine", // le nom de la ligne
                            const int    point_index=0,      // le numéro du point du
                            datetime     time=0,             // la coordonnée du temps
                            double       price=0)            // la coordonnée du prix
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement de la ligne
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction supprime la ligne avec la flèche du graphique.       |
//+------------------------------------------------------------------+
bool ArrowedLineDelete(const long   chart_ID=0,          // ID du graphique
                       const string name="ArrowedLine") // le nom de la ligne
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la ligne avec la flèche
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer la ligne avec la flèche! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement de la ligne, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
void ChangeArrowedLineEmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2)
  {
//--- si le temps du premier point n'est pas spécifié, il sera sur la barre courante
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du deuxième point n'est pas spécifié, il se trouve au 9 barres de ga
   if(!time2)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- établissons le deuxième point au 9 barres de gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas spécifié, il coïncide avec le prix du pre
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de la ligne
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- créons la ligne avec la flèche
   if(!ArrowedLineCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement de la ligne
//--- le compteur du cycle
   int v_steps=accuracy/5;
//--- déplaçons le deuxième point du rattachement selon la verticale
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p2<accuracy-1)
         p2+=1;
      //--- deplaçons le point
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//---déplaçons le premier point du rattachement selon la verticale
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1>1)
         p1-=1;
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- deplaçons le point
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à une demi-seconde
   Sleep(500);
//--- le compteur du cycle
   int h_steps=bars/2;
//--- déplaçons les deux points du rattachement à l'horizontale simultanément
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons les valeurs suivantes
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- deplaçons les points
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.03 seconde
      Sleep(30);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons la ligne avec la flèche
   ArrowedLineDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_CHANNEL

Le canal équidistant.

Note

Pour le canal équidistant on peut spécifier le mode de la suite de son affichage à droite et/ou à gauche (les propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT en conséquence). On peut établir aussi le mode du remplissage du canal par la couleur.

L'exemple

Le script suivant crée et déplace le canal équidistant sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"Le canal équidistant\".
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="Channel";   // Le nom du canal
input int             InpDate1=25;         // La date du 1-ier point en %
input int             InpPrice1=60;        // Le prix du 1-ier point en %
input int             InpDate2=65;         // La date du 2-ième point en %
input int             InpPrice2=80;        // Le prix du 2-ième point en %
input int             InpDate3=30;         // La date du 3-ième point en %
```

© 2000-2015, MetaQuotes Software Corp.

```
input int              InpPrice3=40;         // Le prix du 3-ième point en %
input color            InpColor=clrRed;      // La couleur du canal
input ENUM_LINE_STYLE  InpStyle=STYLE_DASH;  // Le style des lignes du canal
input int              InpWidth=2;           // L'épaisseur des lignes du canal
input bool             InpBack=false;        // Le canal à l'arrière-plan
input bool             InpFill=false;        // Le remplissage du canal par la couleur
input bool             InpSelection=true;    // Sélectionner pour les déplacements
input bool             InpRayLeft=false;     // La suite du canal à gauche
input bool             InpRayRight=false;    // La suite du canal à droite
input bool             InpHidden=true;       // Est caché dans la liste des objets
input long             InpZOrder=0;          // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée le canal équidistant selon les coordonnées spécifiées       |
//+------------------------------------------------------------------+
bool ChannelCreate(const long            chart_ID=0,        // ID du graphique
                   const string          name="Channel",    // le nom du canal
                   const int             sub_window=0,      // le numéro du sous-fenêt
                   datetime              time1=0,           // le temps du premier poi
                   double                price1=0,          // le prix du premier poin
                   datetime              time2=0,           // le temps du deuxième po
                   double                price2=0,          // le prix du deuxième poi
                   datetime              time3=0,           // le temps du troisième p
                   double                price3=0,          // le prix du troisième po
                   const color           clr=clrRed,        // la couleur du canal
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // le style des lignes du
                   const int             width=1,           // l'épaisseur des lignes
                   const bool            fill=false,        // le remplissage du canal
                   const bool            back=false,        // à l'arrière-plan
                   const bool            selection=true,    // sélectionner pour les d
                   const bool            ray_left=false,    // la suite du canal à gau
                   const bool            ray_right=false,   // la suite du canal à dro
                   const bool            hidden=true,       // est caché dans la liste
                   const long            z_order=0)         // la priorité au clic d'u
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le canal selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_CHANNEL,sub_window,time1,price1,time2,price2,tim
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le canal équidistant! Le code de l'erreur = "
      return(false);
     }
//--- établissons la couleur du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- établissons l'épaisseur des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- activons (true) ou désactivons (false) le mode du remplissage du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//---activons (true) ou désactivons (false) le mode de la sélection du canal pour les
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on r
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage du car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage du car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement du canal                        |
//+------------------------------------------------------------------+
bool ChannelPointChange(const long   chart_ID=0,     // ID du graphique
                        const string name="Channel", // le nom du canal
                        const int    point_index=0,  // le numéro du point du rattache
                        datetime     time=0,         // la coordonnée du temps du poin
                        double       price=0)        // la coordonnée du prix du point
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Supprime le canal                                                |
//+------------------------------------------------------------------+
bool ChannelDelete(const long   chart_ID=0,     // ID du graphique
                   const string name="Channel") // le nom du canal
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le canal! Le code de l'erreur = ",GetLast
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement du canal, et pour les valeurs   |
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- si le temps du deuxième point n'est pas spécifié, il sera sur la barre courante
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas spécifié, il aura la valeur Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du premier (gauche) point n'est pas spécifié, il se trouve au 9 barr
   if(!time1)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //---établissons le premier point au 9 barres de gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas spécifié, rapprocherons-le aux 300 points
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si le temps du troisième point n'est pas spécifié, il coïncide avec le temps du
   if(!time3)
      time3=time1;
//--- si le prix du troisième point n'est pas spécifié, il coïncide avec le prix du de
   if(!price3)
      price3=price2;
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- créons le canal équidistant
   if(!ChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
```

© 2000-2015, MetaQuotes Software Corp.

```
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement du canal
//--- le compteur du cycle
   int h_steps=bars/6;
//--- déplaçons le deuxième point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d2<bars-1)
         d2+=1;
      //--- deplaçons le point
      if(!ChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- déplaçons le premier point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d1>1)
         d1-=1;
      //--- deplaçons le point
      if(!ChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
```

© 2000-2015, MetaQuotes Software Corp.

```
   int v_steps=accuracy/10;
//--- déplaçons le troisième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p3>1)
         p3-=1;
      //--- deplaçons le point
      if(!ChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le canal du graphique
   ChannelDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_STDDEVCHANNEL

Le canal de la divergence standard.

Note

Pour le canal de la divergence standard on peut spécifier le mode de la suite de son affichage à droite et/ou à gauche (les propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT en conséquence). On peut établir aussi le mode du remplissage du canal par la couleur.

Pour le changement de la valeur de la divergence du canal on utilise la propriétéOBJPROP_DEVIATION.

L'exemple

Le script suivant crée et déplace le canal de la divergence standard sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"Le canal de la diverger
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="StdDevChannel";   // Le nom du canal
input int             InpDate1=10;               // La date du 1-ier point en %
input int             InpDate2=40;               // La date du 2-ième point en %
input double          InpDeviation=1.0;          // La divergence
```

© 2000-2015, MetaQuotes Software Corp.

```
input color           InpColor=clrRed;           // La couleur du canal
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Le style des lignes du canal
input int             InpWidth=2;                // L'épaisseur des lignes du canal
input bool            InpFill=false;             // Le remplissage du canal par la cou
input bool            InpBack=false;             // Le canal à l'arrière-plan
input bool            InpSelection=true;         // Sélectionner pour les déplacements
input bool            InpRayLeft=false;          // La suite du canal à gauche
input bool            InpRayRight=false;         // La suite du canal à droite
input bool            InpHidden=true;            // Est caché dans la liste des objets
input long            InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée le canal de la divergence standard selon les coordonnées spécifiées      |
//+------------------------------------------------------------------+
bool StdDevChannelCreate(const long            chart_ID=0,        // ID du graphique
                         const string          name="Channel",    // Le nom du canal
                         const int             sub_window=0,      // le numéro du sous
                         datetime              time1=0,           // le temps du premi
                         datetime              time2=0,           // le temps du deuxi
                         const double          deviation=1.0,     // la divergence
                         const color           clr=clrRed,        // La couleur du car
                         const ENUM_LINE_STYLE style=STYLE_SOLID, // Le style des ligr
                         const int             width=1,           //L'épaisseur des li
                         const bool            fill=false,        // Le remplissage du
                         const bool            back=false,        // à l'arrière-plan
                         const bool            selection=true,    // sélectionner pour
                         const bool            ray_left=false,    // la suite du canal
                         const bool            ray_right=false,   // la suite du canal
                         const bool            hidden=true,       // est caché dans la
                         const long            z_order=0)         // la priorité au cl
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeChannelEmptyPoints(time1,time2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le canal selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_STDDEVCHANNEL,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le canal de la divergence standard! Le code d
      return(false);
     }
//--- établissons la valeur de la divergence, la largeur du canal dépend d'elle
   ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation);
//--- établissons la couleur du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- activons (true) ou désactivons (false) le mode du remplissage du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//---activons (true) ou désactivons (false) le mode de la sélection du canal pour les
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on r
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage du car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage du car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement du canal                        |
//+------------------------------------------------------------------+
bool StdDevChannelPointChange(const long   chart_ID=0,     // ID du graphique
                              const string name="Channel", // Le nom du canal
                              const int    point_index=0,  // le numéro du point du ra
                              datetime     time=0)         // la coordonnée du temps d
  {
//--- si le temps du point n'est pas spécifié, déplaçons-le sur la dernière barre
   if(!time)
      time=TimeCurrent();
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change la divergence du canal                                    |
//+------------------------------------------------------------------+
bool StdDevChannelDeviationChange(const long   chart_ID=0,     // ID du graphique
                                  const string name="Channel", // Le nom du canal
```

© 2000-2015, MetaQuotes Software Corp.

```
                                    const double deviation=1.0)  // la divergence
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons l'angle de l'inclinaison de la ligne de la tendance
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la divergence du canal! Le code de l'erreur
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le canal                                                |
//+------------------------------------------------------------------+
bool StdDevChannelDelete(const long   chart_ID=0,     // ID du graphique
                         const string name="Channel") // Le nom du canal
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le canal! Le code de l'erreur = ",GetLast
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement du canal, et pour les valeurs   |
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,datetime &time2)
  {
//--- si le temps du deuxième point n'est pas spécifié, il sera sur la barre courante
   if(!time2)
      time2=TimeCurrent();
//--- si le temps du premier point n'est pas spécifié, il se trouve au 9 barres de gau
   if(!time1)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- établissons le deuxième point au 9 barres de gauche du deuxième
      time1=temp[0];
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
   {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
      {
       Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
       return;
      }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- créons le canal de la divergence standard
   if(!StdDevChannelCreate(0,InpName,0,date[d1],date[d2],InpDeviation,InpColor,InpStyl
      InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder
     {
      return;
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons le canal à l'horizontale à droite et l'élargissons
//--- le compteur du cycle
   int h_steps=bars/2;
//--- déplaçons le canal
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons les valeurs suivantes
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- déplaçons les points du rattachement
      if(!StdDevChannelPointChange(0,InpName,0,date[d1]))
         return;
      if(!StdDevChannelPointChange(0,InpName,1,date[d2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   double v_steps=InpDeviation*2;
//--- élargissons le canal
   for(double i=InpDeviation;i<v_steps;i+=10.0/accuracy)
     {
      if(!StdDevChannelDeviationChange(0,InpName,i))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le canal du graphique
   StdDevChannelDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
```

© 2000-2015, MetaQuotes Software Corp.

```
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_REGRESSION

Le canal sur la régression linéaire.

Note

Pour le canal sur la régression linéaire on peut spécifier le mode de la suite de son affichage à droite et/ou à gauche (les propriétésOBJPROP_RAY_RIGHT et OBJPROP_LEFTen conséquence). On peut établir aussi le mode du remplissage du canal par la couleur.

L'exemple

Le script suivant crée et déplace le canal sur la régression linéaire sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"Le canal sur la régress
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="Regression"; // Le nom du canal
input int             InpDate1=10;          // La date du 1-ier point en %
input int             InpDate2=40;          // La date du 2-ième point en %
input color           InpColor=clrRed;      // La couleur du canal
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Le style des lignes du canal
input int             InpWidth=2;           // L'épaisseur des lignes du canal
```

© 2000-2015, MetaQuotes Software Corp.

```
input bool            InpFill=false;        // Le remplissage du canal par la couleur
input bool            InpBack=false;        // Le canal à l'arrière-plan
input bool            InpSelection=true;    // Sélectionner pour les déplacements
input bool            InpRayLeft=false;     // La suite du canal à gauche
input bool            InpRayRight=false;    // La suite du canal à droite
input bool            InpHidden=true;       // Est caché dans la liste des objets
input long            InpZOrder=0;          // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée le canal sur la régression linéaire selon les coordonnées spécifiées      |
//+------------------------------------------------------------------+
bool RegressionCreate(const long            chart_ID=0,        // ID du graphique
                      const string          name="Regression", // Le nom du canal
                      const int             sub_window=0,      // le numéro du sous-fe
                      datetime              time1=0,           // le temps du premier
                      datetime              time2=0,           // le temps du deuxième
                      const color           clr=clrRed,        // La couleur du canal
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // Le style des lignes
                      const int             width=1,           // L'épaisseur des ligr
                      const bool            fill=false,        // Le remplissage du ca
                      const bool            back=false,        // à l'arrière-plan
                      const bool            selection=true,    // Sélectionner pour le
                      const bool            ray_left=false,    // la suite du canal à
                      const bool            ray_right=false,   // la suite du canal à
                      const bool            hidden=true,       // est caché dans la li
                      const long            z_order=0)         // la priorité au clic
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeRegressionEmptyPoints(time1,time2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le canal selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_REGRESSION,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le canal sur la régression linéaire! Le code
      return(false);
     }
//--- établissons la couleur du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- activons (true) ou désactivons (false) le mode du remplissage du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//---activons (true) ou désactivons (false) le mode de la sélection du canal pour les
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on r
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage du car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage du car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement du canal                        |
//+------------------------------------------------------------------+
bool RegressionPointChange(const long   chart_ID=0,     // ID du graphique
                           const string name="Channel", // Le nom du canal
                           const int    point_index=0,  // le numéro du point du ratta
                           datetime     time=0)         // la coordonnée du temps du p
  {
//--- si le temps du point n'est pas spécifié, déplaçons-le sur la dernière barre
   if(!time)
      time=TimeCurrent();
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le canal                                                |
//+------------------------------------------------------------------+
bool RegressionDelete(const long   chart_ID=0,     // ID du graphique
                      const string name="Channel") // Le nom du canal
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le canal
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le canal! Le code de l'erreur = ",GetLast
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement du canal, et pour les valeurs   |
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeRegressionEmptyPoints(datetime &time1,datetime &time2)
  {
//--- si le temps du deuxième point n'est pas spécifié, il sera sur la barre courante
   if(!time2)
      time2=TimeCurrent();
//--- si le temps du premier point n'est pas spécifié, il se trouve au 9 barres de gau
   if(!time1)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //---établissons le premier point au 9 barres de gauche du deuxième
      time1=temp[0];
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- créons le canal sur la régression linéaire
   if(!RegressionCreate(0,InpName,0,date[d1],date[d2],InpColor,InpStyle,InpWidth,
      InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons le canal à l'horizontale à droite
//--- le compteur du cycle
   int h_steps=bars/2;
//--- déplaçons le canal
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons les valeurs suivantes
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- déplaçons les points du rattachement
      if(!RegressionPointChange(0,InpName,0,date[d1]))
         return;
      if(!RegressionPointChange(0,InpName,1,date[d2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
```

© 2000-2015, MetaQuotes Software Corp.

```
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le canal du graphique
   RegressionDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_PITCHFORK

Andrews' Pitchfork.

Note

Pour "Andrews' Pitchfork" on peut spécifier le mode de son affichage à droite et/ou à gauche (les propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT en conséquence).

On peut aussi spécifier la quantité de lignes-niveau, leurs valeurs et la couleur.

L'exemple

Le script suivant crée et déplace Andrews' Pitchfork sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"Andrews' Pitchfork\"."
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="Pitchfork";  // Le nom de la fourchette
input int             InpDate1=14;          // La date du 1-ier point en %
input int             InpPrice1=40;         // Le prix du 1-ier point en %
input int             InpDate2=18;          // La date du 2-ième point en %
input int             InpPrice2=50;         // Le prix du 2-ième point en %
input int             InpDate3=18;          // La date du 3-ième point en %
input int             InpPrice3=30;         // Le prix du 3-ième point en %
```

© 2000-2015, MetaQuotes Software Corp.

```
input color           InpColor=clrRed;      // La couleur de la fourchette
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // Le style des lignes de la fourchette
input int             InpWidth=1;           // L'épaisseur des lignes de la fourchette
input bool            InpBack=false;        // La fourchette à l'arrière-plan
input bool            InpSelection=true;    // Sélectionner pour les déplacements
input bool            InpRayLeft=false;     // La suite de la fourchette à gauche
input bool            InpRayRight=false;    // La suite de la fourchette à droite
input bool            InpHidden=true;       // Est caché dans la liste des objets
input long            InpZOrder=0;          // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée "Andrews' Pitchfork" selon les coordonnées spécifiées       |
//+------------------------------------------------------------------+
bool PitchforkCreate(const long            chart_ID=0,        // ID du graphique
                     const string          name="Pitchfork",  // le nom de la fourchet
                     const int             sub_window=0,      // le numéro du sous-fen
                     datetime              time1=0,           // le temps du premier p
                     double                price1=0,          // le prix du premier po
                     datetime              time2=0,           // le temps du deuxième
                     double                price2=0,          // le prix du deuxième p
                     datetime              time3=0,           // le temps du troisième
                     double                price3=0,          // le prix du troisième
                     const color           clr=clrRed,        // la couleur des lignes
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // le style des lignes d
                     const int             width=1,           // l'épaisseur des ligne
                     const bool            back=false,        // à l'arrière-plan
                     const bool            selection=true,    // sélectionner pour les
                     const bool            ray_left=false,    // lq suite de lq fourch
                     const bool            ray_right=false,   // la suite de la fourch
                     const bool            hidden=true,       // est caché dans la lis
                     const long            z_order=0)         // la priorité au clic d
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons "Andrews' Pitchfork" selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_PITCHFORK,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"Andrews' Pitchfork\"! Le code de l'erreur =
      return(false);
     }
//--- établissons la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode de la sélection de la fourchette
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage de la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage de la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Spécifie le nombre de niveaux "des Andrews' Pitchfork" et leurs paramètres
//+------------------------------------------------------------------+
bool PitchforkLevelsSet(int             levels,           // le nombre de lignes du ni
                        double          &values[],        // les valeurs des lignes du
                        color           &colors[],        // la couleur des lignes du
                        ENUM_LINE_STYLE &styles[],        // le style des lignes du ni
                        int             &widths[],        // l'épaisseur des lignes du
                        const long      chart_ID=0,       // ID du graphique
                        const string    name="Pitchfork") // le nom de la fourchette
  {
//--- spécifions les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la longueur du tableau ne correspond pas à la quantité de
      return(false);
     }
//--- établissons la quantité de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//---  établissons les propriétés des niveaux dans le cycle
   for(int i=0;i<levels;i++)
     {
      //--- la valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- la couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- le style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- l'épaisseur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- la description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement des "Andrews' Pitchfork"
//+------------------------------------------------------------------+
bool PitchforkPointChange(const long   chart_ID=0,       // ID du graphique
                          const string name="Pitchfork", // le nom du canal
                          const int    point_index=0,    // le numéro du point du ratt
                          datetime     time=0,           // la coordonnée du temps du
                          double       price=0)          // la coordonnée du prix du p
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime "Andrews' Pitchfork"                                    |
//+------------------------------------------------------------------+
bool PitchforkDelete(const long   chart_ID=0,       // ID du graphique
                     const string name="Pitchfork") // Le nom du canal
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"Andrews' Pitchfork\"! Le code de l'erre
      return(false);
     }
//--- l'exécution réussie
   return(true);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points des Andrews' Pitchfork, et pour les valeurs    |
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- si le temps du deuxième (droit supérieur) point n'est pas spécifié, il sera sur
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas spécifié, il aura la valeur Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du premier (gauche) point n'est pas spécifié, il se trouve au 9 barr
   if(!time1)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //---établissons le premier point au 9 barres de gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas spécifié, rapprocherons-le aux 200 points
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si le temps du troisième point n'est pas spécifié, il coïncide avec le temps du
   if(!time3)
      time3=time2;
//--- si le prix du troisième point n'est pas spécifié, rapprocherons-le aux 200 point
   if(!price3)
      price3=price1-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
```

© 2000-2015, MetaQuotes Software Corp.

```
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin des "Andrews' Pitchfork"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- créons la fourchette
   if(!PitchforkCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pric
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement de la fourchette
//--- le compteur du cycle
   int v_steps=accuracy/10;
//--- déplaçons le premier point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1>1)
         p1-=1;
      //--- deplaçons le point
```

© 2000-2015, MetaQuotes Software Corp.

```
      if(!PitchforkPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   int h_steps=bars/8;
//--- déplaçons le troisième point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d3<bars-1)
         d3+=1;
      //--- deplaçons le point
      if(!PitchforkPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   v_steps=accuracy/10;
//--- déplaçons le deuxième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p2>1)
         p2-=1;
      //--- deplaçons le point
      if(!PitchforkPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons la fourchette du graphique
   PitchforkDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_GANNLINE

La ligne de Gann.

Note

Pour la ligne de Gann on peut spécifier le mode de son affichage à droite et/ou à gauche (les propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFTen conséquence).

Pour l'établissement de l'inclinaison de la ligne on peut utiliser l'angle de Gann, ainsi que les coordonnées du deuxième point du rattachement.

L'exemple

Le script suivant crée et déplace la ligne de Gann sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"La ligne de Gann\"."
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="GannLine";        // Le nom de la ligne
input int             InpDate1=20;               // La date du 1-ier point en %
input int             InpPrice1=75;              // Le prix du 1-ier point en %
input int             InpDate2=80;               // La date du 2-ième point en %
input double          InpAngle=0.0;              // L'angle de Gann
```

© 2000-2015, MetaQuotes Software Corp.

```
input double          InpScale=1.0;              // L'échelle
input color           InpColor=clrRed;           // La couleur de la ligne
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Le style de la ligne
input int             InpWidth=2;                // L'épaisseur de la ligne
input bool            InpBack=false;             // La ligne à l'arrière-plan
input bool            InpSelection=true;         // Sélectionner pour les déplacements
input bool            InpRayLeft=false;          // La suite de la ligne à gauche
input bool            InpRayRight=true;          // La suite de la ligne à droite
input bool            InpHidden=true;            // Est caché dans la liste des objets
input long            InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée "la Ligne de Gann" selon les coordonnées, l'angle et l'échelle |
//+------------------------------------------------------------------+
bool GannLineCreate(const long            chart_ID=0,        // ID du graphique
                    const string          name="GannLine",   // le nom de la ligne
                    const int             sub_window=0,      // le numéro du sous-fené
                    datetime              time1=0,           // le temps du premier po
                    double                price1=0,          // le prix du premier poi
                    datetime              time2=0,           // le temps du deuxième p
                    const double          angle=1.0,         // l'angle de Gann
                    const double          scale=1.0,         // l'échelle
                    const color           clr=clrRed,        // la couleur de la ligne
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la ligne
                    const int             width=1,           // l'épaisseur de la ligr
                    const bool            back=false,        // à l'arrière-plan
                    const bool            selection=true,    // sélectionner pour les
                    const bool            ray_left=false,    // la suite de la ligne à
                    const bool            ray_right=true,    // la suite de la ligne à
                    const bool            hidden=true,       // est caché dans la list
                    const long            z_order=0)         // la priorité au clic d'
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeGannLineEmptyPoints(time1,price1,time2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons la ligne de Gann selon les coordonnées spécifiées.
//--- la coordonnée juste du prix du deuxième point du rattachement sera redéfinie
//--- automatiquement après le changement de l'angle de Gann et (ou) de l'échelle,
   if(!ObjectCreate(chart_ID,name,OBJ_GANNLINE,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"La ligne de Gann\"! Le code de l'erreur = "
      return(false);
     }
//--- changeons l'angle de Gann
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- changeons l'échelle (le nombre de pips pour une barre)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- établissons la couleur de la ligne
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de l'affichage de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode de la sélection de la ligne pour
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on r
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage de la
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//---activons (true) ou désactivons (false) le mode de la suite de l'affichage de la l
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement de la ligne de Gann"            |
//+------------------------------------------------------------------+
bool GannLinePointChange(const long   chart_ID=0,      // ID du graphique
                         const string name="GannLine", // le nom de la ligne
                         const int    point_index=0,   // le numéro du point du rattac
                         datetime     time=0,          // la coordonnée du temps du po
                         double       price=0)         // la coordonnée du prix du poi
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement de la ligne
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Change l'angle de Gann                                           |
//+------------------------------------------------------------------+
bool GannLineAngleChange(const long   chart_ID=0,      // ID du graphique
                         const string name="GannLine", // le nom de la ligne
                         const double angle=1.0)       // l'angle de Gann
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons l'angle de Gann
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer l'angle de Gann! Le code de l'erreur = ",Ge
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change l'échelle de l'angle de Gann                              |
//+------------------------------------------------------------------+
bool GannLineScaleChange(const long   chart_ID=0,      // ID du graphique
                         const string name="GannLine", // le nom de la ligne
                         const double scale=1.0)       // l'échelle
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons l'échelle (le nombre de pips pour une barre)
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer l'échelle! Le code de l'erreur = ",GetLastE
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction supprime la ligne de Ganne du graphique.             |
//+------------------------------------------------------------------+
bool GannLineDelete(const long   chart_ID=0,      // ID du graphique
                    const string name="GannLine") // le nom de la ligne
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la ligne de Gann
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"La ligne de Gann\"! Le code de l'erreu
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement de la ligne de Gann, et pour les va
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeGannLineEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- si le temps du deuxième point n'est pas spécifié, il sera sur la barre courante
   if(!time2)
      time2=TimeCurrent();
//--- si le temps du premier point n'est pas spécifié, il se trouve au 9 barres de gau
   if(!time1)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //---établissons le premier point au 9 barres de gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
```

© 2000-2015, MetaQuotes Software Corp.

```
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de la ligne de Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- créons la ligne de Gann.
   if(!GannLineCreate(0,InpName,0,date[d1],price[p1],date[d2],InpAngle,InpScale,InpCol
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons le point du rattachement et changeons l'angle
//--- le compteur du cycle
   int v_steps=accuracy/2;
//---déplaçons le premier point du rattachement selon la verticale
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1>1)
         p1-=1;
      //--- deplaçons le point
      if(!GannLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- le retard à une demi-seconde
   Sleep(500);
//--- définissons une valeur courante de l'angle de Gann (a changé
//--- après le déplacement du premier point du rattachement)
   double curr_angle;
   if(!ObjectGetDouble(0,InpName,OBJPROP_ANGLE,0,curr_angle))
      return;
//--- le compteur du cycle
   v_steps=accuracy/8;
//--- changeons l'angle de Gann
   for(int i=0;i<v_steps;i++)
     {
      if(!GannLineAngleChange(0,InpName,curr_angle-0.05*i))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons la ligne du graphique
   GannLineDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_GANNFAN

Fanion de Gann.

Note

Pour "le fanion de Gann" on peut spécifier le type de la tendance de l'énumération ENUM_GANN_DIRECTION. En réglant la valeur de l'échelle (OBJPROP_SCALE), on peut changer l'angle de l'inclinaison des lignes du fanion.

L'exemple

Le script suivant crée et déplace "le fanion de Gann" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"Fanion de Gann\"."
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="GannFan";         // Le nom du fanion
input int             InpDate1=15;               // La date du 1-ier point en %
input int             InpPrice1=25;              // Le prix du 1-ier point en %
input int             InpDate2=85;               // La date du 2-ième point en %
input double          InpScale=2.0;              // L'échelle
input bool            InpDirection=false;        // La direction de la tendance
```

© 2000-2015, MetaQuotes Software Corp.

```
input color           InpColor=clrRed;           // La couleur du fanion
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Le style des lignes du fanion
input int             InpWidth=1;                // L'épaisseur des lignes du fanion
input bool            InpBack=false;             // Le fanion à l'arrière-plan
input bool            InpSelection=true;         // Sélectionner pour les déplacements
input bool            InpHidden=true;            // Est caché dans la liste des objets
input long            InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée "le Fanion de Gann"                                         |
//+------------------------------------------------------------------+
bool GannFanCreate(const long            chart_ID=0,        // ID du graphique
                   const string          name="GannFan",    // le nom du fanion
                   const int             sub_window=0,      // le numéro du sous-fenêt
                   datetime              time1=0,           // le temps du premier poi
                   double                price1=0,          // le prix du premier poin
                   datetime              time2=0,           // le temps du deuxième po
                   const double          scale=1.0,         // l'échelle
                   const bool            direction=true,    // la direction de la tend
                   const color           clr=clrRed,        // la couleur du fanion
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // le style des lignes du
                   const int             width=1,           // l'épaisseur des lignes
                   const bool            back=false,        // à l'arrière-plan
                   const bool            selection=true,    // sélectionner pour les d
                   const bool            hidden=true,       // est caché dans la liste
                   const long            z_order=0)         // la priorité au clic d'u
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeGannFanEmptyPoints(time1,price1,time2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons "le Fanion de Gann" selon les coordonnées spécifiées.
   if(!ObjectCreate(chart_ID,name,OBJ_GANNFAN,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"la Fanion de Gann\"! Le code de l'erreur =
      return(false);
     }
//--- changeons l'échelle (le nombre de pips pour une barre)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- changeons la direction de la tendance du "Fanion de Gann" (true - descendant, fa
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- établissons la couleur du fanion
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de l'affichage des lignes du fanion
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes du fanion
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- activons (true) ou désactivons (false) le mode de la sélection du fanion pour le
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement du "Fanion de Gann"             |
//+------------------------------------------------------------------+
bool GannFanPointChange(const long   chart_ID=0,     // ID du graphique
                        const string name="GannFan", // le nom du fanion
                        const int    point_index=0,  // le numéro du point du rattache
                        datetime     time=0,         // la coordonnée du temps du poin
                        double       price=0)        // la coordonnée du prix du point
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement du fanion
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change l'échelle du "Fanion de Gann                              |
//+------------------------------------------------------------------+
bool GannFanScaleChange(const long   chart_ID=0,     // ID du graphique
                        const string name="GannFan", // le nom du fanion
                        const double scale=1.0)      // l'échelle
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons l'échelle (le nombre de pips pour une barre)
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer l'échelle! Le code de l'erreur = ",GetLastE
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change la direction de la tendance du "Fanion de Gann"           |
//+------------------------------------------------------------------+
bool GannFanDirectionChange(const long   chart_ID=0,     // ID du graphique
                            const string name="GannFan", // le nom du fanion
                            const bool   direction=true) // la direction de la tendanc
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons la direction de la tendance du "Fanion de Gann"
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la direction de la tendance! Le code de l'e
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction supprime le "Fanion de Ganne" du graphique.          |
//+------------------------------------------------------------------+
bool GannFanDelete(const long   chart_ID=0,     // ID du graphique
                   const string name="GannFan") // le nom du fanion
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le fanion de Gann
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"la Fanion de Gann\"! Le code de l'erreu
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement du "Fanion de Gann" et pour les val
//| vides établit les valeurs par défaut                             |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
void ChangeGannFanEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- si le temps du deuxième point n'est pas spécifié, il sera sur la barre courante
   if(!time2)
      time2=TimeCurrent();
//--- si le temps du premier point n'est pas spécifié, il se trouve au 9 barres de gau
   if(!time1)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //---établissons le premier point au 9 barres de gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du fanion de Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- créons le fanion de Gann
   if(!GannFanCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons le point du rattachement du fanion
//--- le compteur du cycle
   int v_steps=accuracy/2;
//---déplaçons le premier point du rattachement selon la verticale
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1<accuracy-1)
         p1+=1;
      //--- deplaçons le point
      if(!GannFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- changeons la direction de la tendance du fanion au descendant
   GannFanDirectionChange(0,InpName,true);
//--- redessinons le graphique
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le fanion du graphique
   GannFanDelete(0,InpName);
```

© 2000-2015, MetaQuotes Software Corp.

```
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_GANNGRID

Grille de Gann.

Note

Pour "Grille de Gann" on peut spécifier le type de la tendance de l'énumération ENUM_GANN_DIRECTION. En réglant la valeur de l'échelle (OBJPROP_SCALE), on peut changer l'angle de l'inclinaison des lignes de la grille.

L'exemple

Le script suivant crée et déplace "la Grille de Gann" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"Grille de Gann\"."
#property description "Les coordonnées des points du rattachement de la grille sont sp
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="GannGrid";        // Le nom de la grille
input int             InpDate1=15;               // La date du 1-ier point en %
input int             InpPrice1=25;              // Le prix du 1-ier point en %
input int             InpDate2=35;               // La date du 2-ième point en %
input double          InpScale=3.0;              // L'échelle
input bool            InpDirection=false;        // La direction de la tendance
```

© 2000-2015, MetaQuotes Software Corp.

```
input color           InpColor=clrRed;           // La couleur de la grille
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Le style des lignes de la grille
input int             InpWidth=1;                // L'épaisseur des lignes du fanion
input bool            InpBack=false;             // La grille à l'arrière-plan
input bool            InpSelection=true;         // Sélectionner pour les déplacements
input bool            InpHidden=true;            // Est caché dans la liste des objets
input long            InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée "la Grille de Gann"                                         |
//+------------------------------------------------------------------+
bool GannGridCreate(const long            chart_ID=0,        // ID du graphique
                    const string          name="GannGrid",   // le nom de la grille
                    const int             sub_window=0,      // le numéro du sous-fené
                    datetime              time1=0,           // le temps du premier po
                    double                price1=0,          // le prix du premier poi
                    datetime              time2=0,           // le temps du deuxième p
                    const double          scale=1.0,         // l'échelle
                    const bool            direction=true,    // la direction de la ten
                    const color           clr=clrRed,        // la couleur de la grill
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // le style des lignes de
                    const int             width=1,           // l'épaisseur des lignes
                    const bool            back=false,        // à l'arrière-plan
                    const bool            selection=true,    // sélectionner pour les
                    const bool            hidden=true,       // est caché dans la list
                    const long            z_order=0)         // la priorité au clic d'
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeGannGridEmptyPoints(time1,price1,time2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons "la Grille de Gann" selon les coordonnées spécifiées.
   if(!ObjectCreate(chart_ID,name,OBJ_GANNGRID,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"la Grille de Gann\"! Le code de l'erreur =
      return(false);
     }
//--- changeons l'échelle (le nombre de pips pour une barre)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- changeons la direction de la tendance de "la Grille de Gann" (true - descendant,
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- établissons la couleur de la grille
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de l'affichage des lignes de la grille
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes de la grille
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- activons (true) ou désactivons (false) le mode de la sélection de la grille pour
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement de "la Grille de Gann"          |
//+------------------------------------------------------------------+
bool GannGridPointChange(const long   chart_ID=0,      // ID du graphique
                         const string name="GannGrid", // le nom de la grille
                         const int    point_index=0,   // le numéro du point du rattac
                         datetime     time=0,          // la coordonnée du temps du po
                         double       price=0)         // la coordonnée du prix du poi
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement de la grille
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change l'échelle de "la Grille de Gann                           |
//+------------------------------------------------------------------+
bool GannGridScaleChange(const long   chart_ID=0,      // ID du graphique
                         const string name="GannGrid", // le nom de l'objet "la Grille
                         const double scale=1.0)       // l'échelle
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons l'échelle (le nombre de pips pour une barre)
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer l'échelle! Le code de l'erreur = ",GetLastE
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change la direction de la tendance de "la Grille de Gann"        |
//+------------------------------------------------------------------+
bool GannGridDirectionChange(const long   chart_ID=0,      // ID du graphique
                             const string name="GannGrid", // le nom de la grille
                             const bool   direction=true)  // la direction de la tenda
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons la direction de la tendance de "la Grille de Gann"
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la direction de la tendance! Le code de l'e
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction supprime la "Grille de Ganne" du graphique           |
//+------------------------------------------------------------------+
bool GannGridDelete(const long   chart_ID=0,      // ID du graphique
                    const string name="GannGrid") // le nom de la grille
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la Grille de Gann
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"la Grille de Gann\"! Le code de l'erreu
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement de "la Grille de Gann" et pour les
//| vides établit les valeurs par défaut                             |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
void ChangeGannGridEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- si le temps du deuxième point n'est pas spécifié, il sera sur la barre courante
   if(!time2)
      time2=TimeCurrent();
//--- si le temps du premier point n'est pas spécifié, il se trouve au 9 barres de gau
   if(!time1)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //---établissons le premier point au 9 barres de gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de la grille de Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- créons la grille de Gann
   if(!GannGridCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement de la grille
//--- le compteur du cycle
   int v_steps=accuracy/4;
//---déplaçons le premier point du rattachement selon la verticale
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1<accuracy-1)
         p1+=1;
      if(!GannGridPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   int h_steps=bars/4;
//--- déplaçons le deuxième point du rattachement selon l'horizontale
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d2<bars-1)
         d2+=1;
      if(!GannGridPointChange(0,InpName,1,date[d2],0))
```

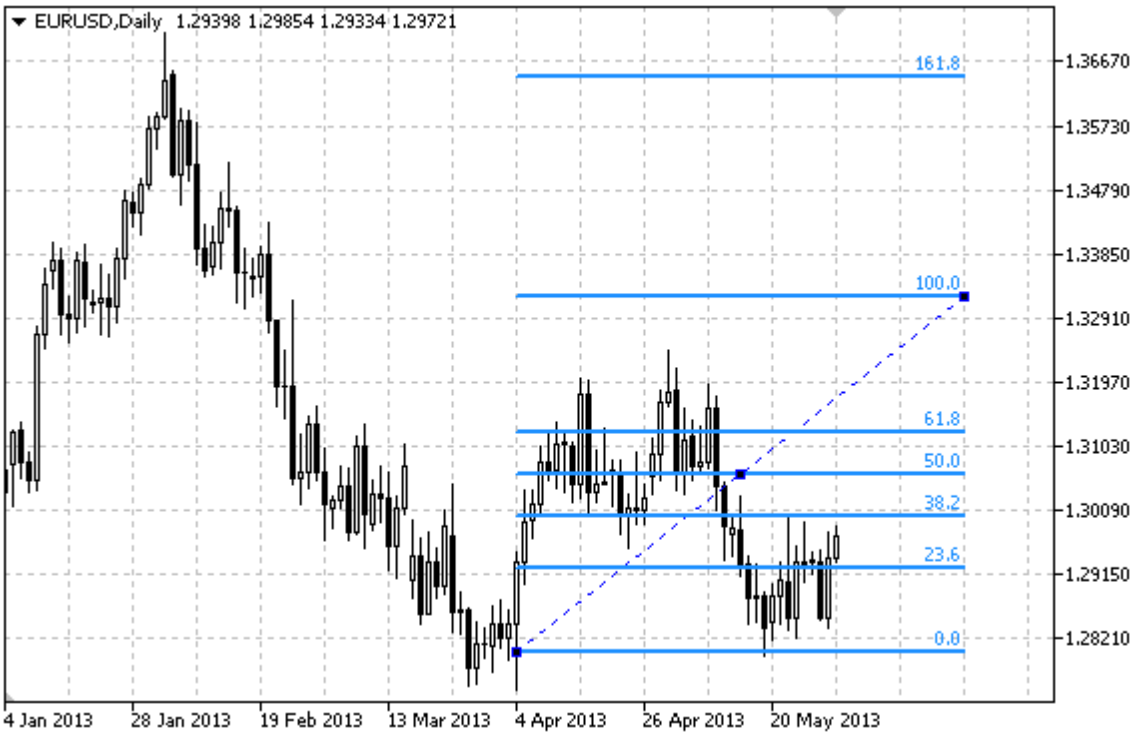
© 2000-2015, MetaQuotes Software Corp.

```
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//---changeons la direction de la tendance de la grille au descendant
   GannGridDirectionChange(0,InpName,true);
//--- redessinons le graphique
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons la grille du graphique
   GannGridDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_FIBO

Les niveaux de Fibonacci.



Note

Pour "Les niveaux de Fibonacci" on peut spécifier le mode de son affichage à droite et/ou à gauche (les propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT en conséquence).

On peut aussi spécifier la quantité de lignes-niveau, leurs valeurs et la couleur.

L'exemple

Le script suivant crée et déplace "Les niveaux de Fibonacci" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"Les niveaux de Fibonacc
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="FiboLevels";      // Le nom de l'objet
input int             InpDate1=10;               // La date du 1-ier point en %
input int             InpPrice1=65;              // Le prix du 1-ier point en %
input int             InpDate2=90;               // La date du 2-ième point en %
input int             InpPrice2=85;              // Le prix du 2-ième point en %
input color           InpColor=clrRed;           // La couleur de l'objet
```

© 2000-2015, MetaQuotes Software Corp.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Le style de la ligne
input int             InpWidth=2;                // L'épaisseur de la ligne
input bool            InpBack=false;             // L'objet à l'arrière-plan
input bool            InpSelection=true;         // Sélectionner pour les déplacements
input bool            InpRayLeft=false;          // La suite de l'objet à gauche
input bool            InpRayRight=false;         // La suite de l'objet à droite
input bool            InpHidden=true;            // Est caché dans la liste des objets
input long            InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée "Les niveaux de Fibonacci" selon les coordonnées spécifiées |
//+------------------------------------------------------------------+
bool FiboLevelsCreate(const long            chart_ID=0,        // ID du graphique
                      const string          name="FiboLevels", // le nom de l'objet
                      const int             sub_window=0,      // le numéro du sous-fe
                      datetime              time1=0,           // le temps du premier
                      double                price1=0,          // le prix du premier p
                      datetime              time2=0,           // le temps du deuxième
                      double                price2=0,          // le prix du deuxième
                      const color           clr=clrRed,        // la couleur de l'obje
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la ligne
                      const int             width=1,           // l'épaisseur de la li
                      const bool            back=false,        // à l'arrière-plan
                      const bool            selection=true,    // Sélectionner pour le
                      const bool            ray_left=false,    // la suite de l'objet
                      const bool            ray_right=false,   // la suite de l'objet
                      const bool            hidden=true,       // est caché dans la li
                      const long            z_order=0)         // la priorité au clic
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeFiboLevelsEmptyPoints(time1,price1,time2,price2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons "Les niveaux de Fibonacci" selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_FIBO,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"Les niveaux de Fibonacci\"! Le code de l'er
      return(false);
     }
//--- établissons la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode de la sélection de l'objet pour l
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage de l'o
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage de l'o
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Spécifie le nombre de niveaux et leurs paramètres                |
//+------------------------------------------------------------------+
bool FiboLevelsSet(int              levels,           // le nombre de lignes du niveau
                   double           &values[],        // les valeurs des lignes du niv
                   color            &colors[],        // la couleur des lignes du nive
                   ENUM_LINE_STYLE &styles[],         // le style des lignes du niveau
                   int              &widths[],        // l'épaisseur des lignes du niv
                   const long       chart_ID=0,       // ID du graphique
                   const string     name="FiboLevels") // le nom de l'objet
  {
//--- spécifions les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la longueur du tableau ne correspond pas à la quantité de
      return(false);
     }
//--- établissons la quantité de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//---  établissons les propriétés des niveaux dans le cycle
   for(int i=0;i<levels;i++)
     {
      //--- la valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- la couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- le style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- l'épaisseur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- la description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement des "Niveaux de Fibonacci"      |
//+------------------------------------------------------------------+
bool FiboLevelsPointChange(const long   chart_ID=0,        // ID du graphique
                           const string name="FiboLevels", // le nom de l'objet
                           const int    point_index=0,     // le numéro du point du ra
                           datetime     time=0,            // la coordonnée du temps d
                           double       price=0)           // la coordonnée du prix du
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime "Les niveaux de Fibonacci"                              |
//+------------------------------------------------------------------+
bool FiboLevelsDelete(const long   chart_ID=0,        // ID du graphique
                      const string name="FiboLevels") // le nom de l'objet
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"Les niveaux de Fibonacci\"! Le code de
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement des "Niveaux de Fibonacci" et
```

© 2000-2015, MetaQuotes Software Corp.

```
//| pour les valeurs établit les valeurs par défaut                  |
//+------------------------------------------------------------------+
void ChangeFiboLevelsEmptyPoints(datetime &time1,double &price1,
                                 datetime &time2,double &price2)
  {
//--- si le temps du deuxième point n'est pas spécifié, il sera sur la barre courante
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas spécifié, il aura la valeur Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du premier point n'est pas spécifié, il se trouve au 9 barres de gau
   if(!time1)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //---établissons le premier point au 9 barres de gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas spécifié, rapprocherons-le aux 200 points
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin des "Niveaux de Fibonacci"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- créons l'objet
   if(!FiboLevelsCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement
//--- le compteur du cycle
   int v_steps=accuracy*2/5;
//--- déplaçons le premier point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1>1)
         p1-=1;
      //--- deplaçons le point
      if(!FiboLevelsPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   v_steps=accuracy*4/5;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- déplaçons le deuxième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p2>1)
         p2-=1;
      //--- deplaçons le point
      if(!FiboLevelsPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons l'objet du graphique
   FiboLevelsDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_FIBOTIMES

Les zones temporaires de Fibonacci.

Note

Pour"Les zones temporaires de Fibonacci" on peut spécifier la quantité de lignes-niveau, leurs valeurs et la couleur.

L'exemple

Le script suivant crée et déplace "Les zones temporaires de Fibonacci" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"Les zones temporaires d
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="FiboTimes";       // Le nom de l'objet
input int             InpDate1=10;               // La date du 1-ier point en %
input int             InpPrice1=45;              // Le prix du 1-ier point en %
input int             InpDate2=20;               // La date du 2-ième point en %
input int             InpPrice2=55;              // Le prix du 2-ième point en %
input color           InpColor=clrRed;           // La couleur de l'objet
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Le style de la ligne
input int             InpWidth=2;                // L'épaisseur de la ligne
```

© 2000-2015, MetaQuotes Software Corp.

```
input bool             InpBack=false;             // L'objet à l'arrière-plan
input bool             InpSelection=true;         // Sélectionner pour les déplacements
input bool             InpHidden=true;            // Est caché dans la liste des objets
input long             InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée "Les zones temporaires de Fibonacci" selon les coordonnées spécifiées |
//+------------------------------------------------------------------+
bool FiboTimesCreate(const long            chart_ID=0,        // ID du graphique
                     const string          name="FiboTimes",  // le nom de l'objet
                     const int             sub_window=0,      // le numéro du sous-fen
                     datetime              time1=0,           // le temps du premier p
                     double                price1=0,          // le prix du premier po
                     datetime              time2=0,           // le temps du deuxième
                     double                price2=0,          // le prix du deuxième p
                     const color           clr=clrRed,        // la couleur de l'objet
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la ligne
                     const int             width=1,           // l'épaisseur de la lig
                     const bool            back=false,        // à l'arrière-plan
                     const bool            selection=true,    // sélectionner pour les
                     const bool            hidden=true,       // est caché dans la lis
                     const long            z_order=0)         // la priorité au clic d
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeFiboTimesEmptyPoints(time1,price1,time2,price2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons "Les zones temporaires de Fibonacci" selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOTIMES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"Les zones temporaires de Fibonacci\"! Le co
      return(false);
     }
//--- établissons la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode de la sélection de l'objet pour l
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2015, MetaQuotes Software Corp.

```
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Spécifie le nombre de niveaux et leurs paramètres                |
//+------------------------------------------------------------------+
bool FiboTimesLevelsSet(int             levels,           // le nombre de lignes du ni
                        double          &values[],        // les valeurs des lignes du
                        color           &colors[],        // la couleur des lignes du
                        ENUM_LINE_STYLE &styles[],        // le style des lignes du ni
                        int             &widths[],        // l'épaisseur des lignes du
                        const long      chart_ID=0,       // ID du graphique
                        const string    name="FiboTimes") // le nom de l'objet
  {
//--- spécifions les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la longueur du tableau ne correspond pas à la quantité de
      return(false);
     }
//--- établissons la quantité de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//---  établissons les propriétés des niveaux dans le cycle
   for(int i=0;i<levels;i++)
     {
      //--- la valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- la couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- le style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- l'épaisseur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- la description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(values[i],1));
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement des "Zones temporaires de Fibonacci"
//+------------------------------------------------------------------+
bool FiboTimesPointChange(const long   chart_ID=0,       // ID du graphique
                          const string name="FiboTimes", // le nom de l'objet
                          const int    point_index=0,    // le numéro du point du ratt
                          datetime     time=0,           // la coordonnée du temps du
```

© 2000-2015, MetaQuotes Software Corp.

```
                           double       price=0)          // la coordonnée du prix du p
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'erı
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime "Les zones temporaires de Fibonacci"                    |
//+------------------------------------------------------------------+
bool FiboTimesDelete(const long   chart_ID=0,       // ID du graphique
                     const string name="FiboTimes") // le nom de l'objet
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"Les zones temporaires de Fibonacci\"! I
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement des "Zones temporaires de Fibonacci
//| pour les valeurs vides établit les valeurs par défaut            |
//+------------------------------------------------------------------+
void ChangeFiboTimesEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- si le temps du premier point n'est pas spécifié, il sera sur la barre courante
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
```

© 2000-2015, MetaQuotes Software Corp.

```
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du deuxième point n'est pas spécifié, il se trouve au 2 barres de ga
   if(!time2)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 3 dernières barre
      datetime temp[3];
      CopyTime(Symbol(),Period(),time1,3,temp);
      //--- établissons le premier point au 2 barres de gauche du deuxième
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas spécifié, il coïncide avec le prix du pre
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
```

© 2000-2015, MetaQuotes Software Corp.

```
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin des "Zones temporaires de Fibonacci"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- créons l'objet
   if(!FiboTimesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement
//--- le compteur du cycle
   int h_steps=bars*2/5;
//--- déplaçons le deuxième point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d2<bars-1)
         d2+=1;
      //--- deplaçons le point
      if(!FiboTimesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   h_steps=bars*3/5;
//--- déplaçons le premier point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d1<bars-1)
         d1+=1;
      //--- deplaçons le point
      if(!FiboTimesPointChange(0,InpName,0,date[d1],price[p1]))
```

© 2000-2015, MetaQuotes Software Corp.

```
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons l'objet du graphique
   FiboTimesDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_FIBOFAN

Le fanion de Fibonacci.

Note

On peut spécifier la quantité de lignes-niveau, leurs valeurs et la couleurpour"Le fanion de Fibonacci".

L'exemple

Le script suivant crée et déplace "Le fanion de Fibonacci" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"Le fanion de Fibonacci\
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="FiboFan";         // Le nom du fanion
input int             InpDate1=10;               // La date du 1-ier point en %
input int             InpPrice1=25;              // Le prix du 1-ier point en %
input int             InpDate2=30;               // La date du 2-ième point en %
input int             InpPrice2=50;              // Le prix du 2-ième point en %
input color           InpColor=clrRed;           // La couleur de la ligne du fanion
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Le style de la ligne
```

© 2000-2015, MetaQuotes Software Corp.

```
input int              InpWidth=2;                // L'épaisseur de la ligne
input bool             InpBack=false;             // L'objet à l'arrière-plan
input bool             InpSelection=true;         // Sélectionner pour les déplacements
input bool             InpHidden=true;            // Est caché dans la liste des objets
input long             InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée "Le fanion de Fibonacci" selon les coordonnées spécifiées   |
//+------------------------------------------------------------------+
bool FiboFanCreate(const long            chart_ID=0,        // ID du graphique
                   const string          name="FiboFan",    // le nom du fanion
                   const int             sub_window=0,      // le numéro du sous-fenêt
                   datetime              time1=0,           // le temps du premier poi
                   double                price1=0,          // le prix du premier poin
                   datetime              time2=0,           // le temps du deuxième po
                   double                price2=0,          // le prix du deuxième poi
                   const color           clr=clrRed,        // la couleur de la ligne
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la ligne du
                   const int             width=1,           // l'épaisseur de la ligne
                   const bool            back=false,        // à l'arrière-plan
                   const bool            selection=true,    // sélectionner pour les d
                   const bool            hidden=true,       // est caché dans la liste
                   const long            z_order=0)         // la priorité au clic d'u
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeFiboFanEmptyPoints(time1,price1,time2,price2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons "Le fanion de Fibonacci" selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOFAN,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"Le fanion de Fibonacci.\"! Le code de l'err
      return(false);
     }
//--- établissons la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode de la sélection du fanion pour le
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Spécifie le nombre de niveaux et leurs paramètres                |
//+------------------------------------------------------------------+
bool FiboFanLevelsSet(int             levels,         // le nombre de lignes du niveau
                      double          &values[],      // les valeurs des lignes du niv
                      color           &colors[],      // la couleur des lignes du nive
                      ENUM_LINE_STYLE &styles[],      // le style des lignes du niveau
                      int             &widths[],      // l'épaisseur des lignes du niv
                      const long      chart_ID=0,     // ID du graphique
                      const string    name="FiboFan") // le nom du fanion
  {
//--- spécifions les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la longueur du tableau ne correspond pas à la quantité de
      return(false);
     }
//--- établissons la quantité de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//---  établissons les propriétés des niveaux dans le cycle
   for(int i=0;i<levels;i++)
     {
      //--- la valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- la couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- le style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- l'épaisseur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- la description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement du "Fanion de Fibonacci"        |
//+------------------------------------------------------------------+
bool FiboFanPointChange(const long   chart_ID=0,     // ID du graphique
                        const string name="FiboFan", // le nom du fanion
                        const int    point_index=0,  // le numéro du point du rattache
```

© 2000-2015, MetaQuotes Software Corp.

```
                        datetime     time=0,         // la coordonnée du temps du poi
                        double       price=0)        // la coordonnée du prix du point
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime "Le fanion de Fibonacci"                                |
//+------------------------------------------------------------------+
bool FiboFanDelete(const long   chart_ID=0,     // ID du graphique
                   const string name="FiboFan") // le nom du fanion
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le fanion
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"Le fanion de Fibonacci\"! Le code de l'
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement du "Fanion de Fibonacci", et
//| pour les valeurs établit les valeurs par défaut                  |
//+------------------------------------------------------------------+
void ChangeFiboFanEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- si le temps du deuxième point n'est pas spécifié, il sera sur la barre courante
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas spécifié, il aura la valeur Bid
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du premier point n'est pas spécifié, il se trouve au 9 barres de gau
   if(!time1)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //---établissons le premier point au 9 barres de gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas spécifié, rapprocherons-le aux 200 points
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du "Fanion de Fibonacci"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- créons l'objet
   if(!FiboFanCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//---maintenant déplaçons les points du fanion
//--- le compteur du cycle
   int v_steps=accuracy/2;
//--- déplaçons le premier point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1<accuracy-1)
         p1+=1;
      //--- deplaçons le point
      if(!FiboFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   int h_steps=bars/4;
//--- déplaçons le deuxième point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d2<bars-1)
         d2+=1;
      //--- deplaçons le point
      if(!FiboFanPointChange(0,InpName,1,date[d2],price[p2]))
         return;
```

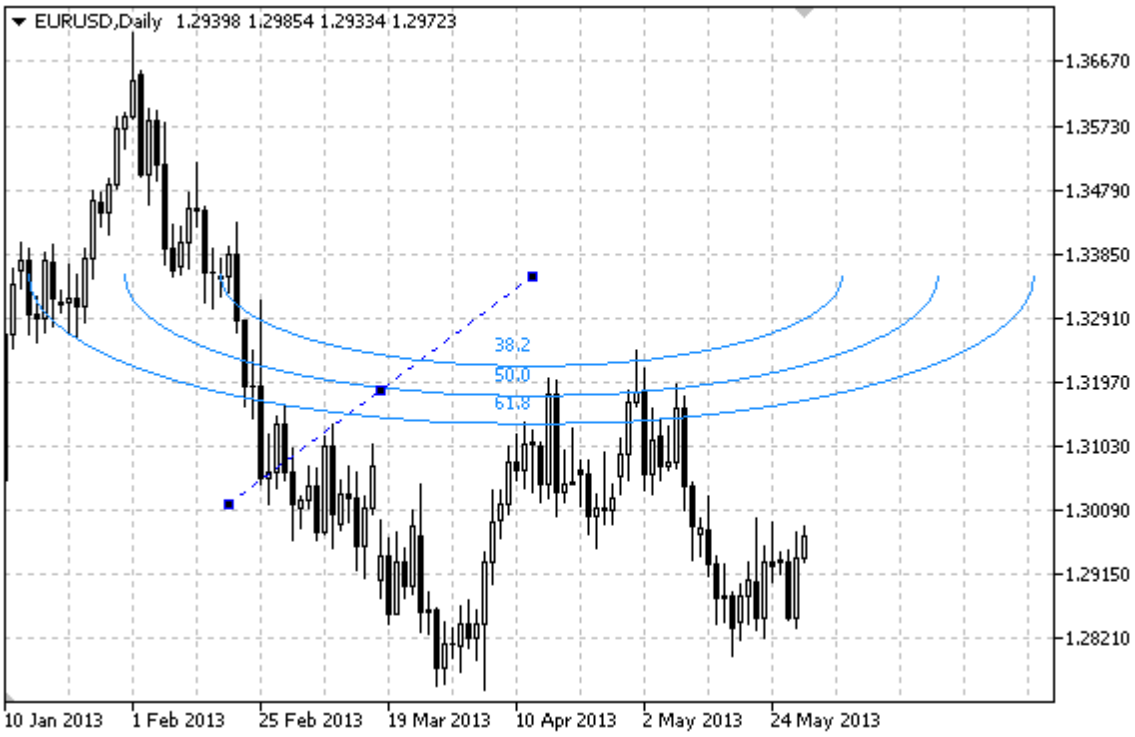
© 2000-2015, MetaQuotes Software Corp.

```
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons l'objet du graphique
   FiboFanDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_FIBOARC

Les arcs de Fibonacci.



Note

Pour"Les arcs de Fibonacci" on peut spécifier le mode de l'affichage de toute l'ellipse entièrement. On peut spécifier le rayon de la courbure des lignes en changeant l'échelle et les coordonnées des points du rattachement.

On peut aussi spécifier la quantité de lignes-niveau, leurs valeurs et la couleur.

L'exemple

Le script suivant crée et déplace "Les arcs de Fibonacci" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"Les arcs de Fibonacci\"
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="FiboArc";         // Le nom de l'objet
input int             InpDate1=25;               // La date du 1-ier point en %
input int             InpPrice1=25;              // Le prix du 1-ier point en %
input int             InpDate2=35;               // La date du 2-ième point en %
input int             InpPrice2=55;              // Le prix du 2-ième point en %
```

© 2000-2015, MetaQuotes Software Corp.

```
input double          InpScale=3.0;              // L'échelle
input bool            InpFullEllipse=true;       // La forme des arcs
input color           InpColor=clrRed;           // La couleur de la ligne
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Le style de la ligne
input int             InpWidth=2;                // L'épaisseur de la ligne
input bool            InpBack=false;             // L'objet à l'arrière-plan
input bool            InpSelection=true;         // Sélectionner pour les déplacements
input bool            InpHidden=true;            // Est caché dans la liste des objets
input long            InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée "Les arcs de Fibonacci" selon les coordonnées spécifiées    |
//+------------------------------------------------------------------+
bool FiboArcCreate(const long            chart_ID=0,         // ID du graphique
                   const string          name="FiboArc",     // le nom de l'objet
                   const int             sub_window=0,       // le numéro du sous-fené
                   datetime              time1=0,            // le temps du premier po
                   double                price1=0,           // le prix du premier poi
                   datetime              time2=0,            // le temps du deuxième p
                   double                price2=0,           // le prix du deuxième po
                   const double          scale=1.0,          // l'échelle
                   const bool            full_ellipse=false, // la forme des arcs
                   const color           clr=clrRed,         // la couleur de la ligne
                   const ENUM_LINE_STYLE style=STYLE_SOLID,  // le style de la ligne
                   const int             width=1,            // l'épaisseur de la lig
                   const bool            back=false,         // à l'arrière-plan
                   const bool            selection=true,     // Sélectionner pour les
                   const bool            hidden=true,        // est caché dans la list
                   const long            z_order=0)          // la priorité au clic d'
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeFiboArcEmptyPoints(time1,price1,time2,price2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons "Les arcs de Fibonacci" selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOARC,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"Les arcs de Fibonacci\"! Le code de l'erreu
      return(false);
     }
//--- établissons l'échelle
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//---établissons l'affichage des arcs en forme de l'ellipse complète (true) ou de la m
   ObjectSetInteger(chart_ID,name,OBJPROP_ELLIPSE,full_ellipse);
//--- établissons la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de la ligne
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur de la ligne
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode de la sélection des arcs pour les
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Spécifie le nombre de niveaux et leurs paramètres                |
//+------------------------------------------------------------------+
bool FiboArcLevelsSet(int             levels,         // le nombre de lignes du niveau
                      double          &values[],      // les valeurs des lignes du niv
                      color           &colors[],      // la couleur des lignes du nive
                      ENUM_LINE_STYLE &styles[],      // le style des lignes du niveau
                      int             &widths[],      // l'épaisseur des lignes du niv
                      const long      chart_ID=0,     // ID du graphique
                      const string    name="FiboArc") // le nom de l'objet
  {
//--- spécifions les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la longueur du tableau ne correspond pas à la quantité de
      return(false);
     }
//--- établissons la quantité de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//---  établissons les propriétés des niveaux dans le cycle
   for(int i=0;i<levels;i++)
     {
      //--- la valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- la couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- le style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- l'épaisseur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- la description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement des "Arcs de Fibonacci"         |
//+------------------------------------------------------------------+
bool FiboArcPointChange(const long   chart_ID=0,     // ID du graphique
                        const string name="FiboArc", // le nom de l'objet
                        const int    point_index=0,  // le numéro du point du rattache
                        datetime     time=0,         // la coordonnée du temps du poin
                        double       price=0)        // la coordonnée du prix du point
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime "Les arcs de Fibonacci"                                 |
//+------------------------------------------------------------------+
bool FiboArcDelete(const long   chart_ID=0,     // ID du graphique
                   const string name="FiboArc") // le nom de l'objet
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"Les arcs de Fibonacci\"! Le code de l'e
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Vérifie les valeurs des points du rattachement des "Arcs de Fibonacci" et pour les
//| vides établit les valeurs par défaut                              |
//+------------------------------------------------------------------+
void ChangeFiboArcEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- si le temps du deuxième point n'est pas spécifié, il sera sur la barre courante
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas spécifié, il aura la valeur Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du premier point n'est pas spécifié, il se trouve au 9 barres de gau
   if(!time1)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //---établissons le premier point au 9 barres de gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas spécifié, rapprocherons-le aux 300 points
   if(!price1)
      price1=price2-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
```

© 2000-2015, MetaQuotes Software Corp.

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin des "Arcs de Fibonacci"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- créons l'objet
   if(!FiboArcCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpScale,
      InpFullEllipse,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrd
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement
//--- le compteur du cycle
   int v_steps=accuracy/5;
//--- déplaçons le premier point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1<accuracy-1)
         p1+=1;
      //--- deplaçons le point
      if(!FiboArcPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
```

© 2000-2015, MetaQuotes Software Corp.

```
   int h_steps=bars/5;
//--- déplaçons le deuxième point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d2<bars-1)
         d2+=1;
      //--- deplaçons le point
      if(!FiboArcPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons l'objet du graphique
   FiboArcDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_FIBOCHANNEL

Le canal de Fibonacci.

Note

Pour "Le canal de Fibonacci" on peut spécifier le mode de la suite de son affichage à droite et/ou à gauche (les propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT en conséquence).

On peut aussi spécifier la quantité de lignes-niveau, leurs valeurs et la couleur.

L'exemple

Le script suivant crée et déplace "Le canal de Fibonacci" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"Le canal de Fibonacci\"
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="FiboChannel";     // Le nom du canal
input int             InpDate1=20;               // La date du 1-ier point en %
input int             InpPrice1=10;              // Le prix du 1-ier point en %
input int             InpDate2=60;               // La date du 2-ième point en %
input int             InpPrice2=30;              // Le prix du 2-ième point en %
input int             InpDate3=20;               // La date du 3-ième point en %
```

© 2000-2015, MetaQuotes Software Corp.

```
input int              InpPrice3=25;              // Le prix du 3-ième point en %
input color            InpColor=clrRed;           // La couleur du canal
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Le style des lignes du canal
input int              InpWidth=2;                // L'épaisseur des lignes du canal
input bool             InpBack=false;             // Le canal à l'arrière-plan
input bool             InpSelection=true;         // Sélectionner pour les déplacements
input bool             InpRayLeft=false;          // La suite du canal à gauche
input bool             InpRayRight=false;         // La suite du canal à droite
input bool             InpHidden=true;            // Est caché dans la liste des objets
input long             InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée "Le canal de Fibonacci" selon les coordonnées spécifiées    |
//+------------------------------------------------------------------+
bool FiboChannelCreate(const long             chart_ID=0,         // ID du graphique
                       const string           name="FiboChannel", // Le nom du canal
                       const int              sub_window=0,       // le numéro du sous-
                       datetime               time1=0,            // le temps du premie
                       double                 price1=0,           // le prix du premier
                       datetime               time2=0,            // le temps du deuxié
                       double                 price2=0,           // le prix du deuxièm
                       datetime               time3=0,            // le temps du troisi
                       double                 price3=0,           // le prix du troisié
                       const color            clr=clrRed,         // La couleur du cana
                       const ENUM_LINE_STYLE  style=STYLE_SOLID,  // Le style des ligne
                       const int              width=1,            // L'épaisseur des li
                       const bool             back=false,         // à l'arrière-plan
                       const bool             selection=true,     // sélectionner pour
                       const bool             ray_left=false,     // la suite du canal
                       const bool             ray_right=false,    // la suite du canal
                       const bool             hidden=true,        // est caché dans la
                       const long             z_order=0)          // la priorité au cli
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeFiboChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le canal selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOCHANNEL,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"Le canal de Fibonacci\"! Le code de l'erreu
      return(false);
     }
//--- établissons la couleur du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes du canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//---activons (true) ou désactivons (false) le mode de la sélection du canal pour les
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on r
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage du car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage du car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Spécifie le nombre de niveaux et leurs paramètres                |
//+------------------------------------------------------------------+
bool FiboChannelLevelsSet(int             levels,             // le nombre de lignes d
                          double          &values[],          // les valeurs des ligne
                          color           &colors[],          // la couleur des lignes
                          ENUM_LINE_STYLE &styles[],          // le style des lignes d
                          int             &widths[],          // l'épaisseur des ligne
                          const long      chart_ID=0,         // ID du graphique
                          const string    name="FiboChannel") // le nom de l'objet
  {
//--- spécifions les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la longueur du tableau ne correspond pas à la quantité de
      return(false);
     }
//--- établissons la quantité de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//---  établissons les propriétés des niveaux dans le cycle
   for(int i=0;i<levels;i++)
     {
      //--- la valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- la couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- le style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- l'épaisseur du niveau
```

© 2000-2015, MetaQuotes Software Corp.

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- la description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement du "Canal de Fibonacci"         |
//+------------------------------------------------------------------+
bool FiboChannelPointChange(const long   chart_ID=0,         // ID du graphique
                            const string name="FiboChannel", // Le nom du canal
                            const int    point_index=0,      // le numéro du point du
                            datetime     time=0,             // la coordonnée du temps
                            double       price=0)            // la coordonnée du prix
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le canal                                                |
//+------------------------------------------------------------------+
bool FiboChannelDelete(const long   chart_ID=0,         // ID du graphique
                       const string name="FiboChannel") // Le nom du canal
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"Le canal de Fibonacci\"! Le code de l'e
      return(false);
     }
//--- l'exécution réussie
```

© 2000-2015, MetaQuotes Software Corp.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement du "Canal de Fibonacci", et      |
//| pour les valeurs établit les valeurs par défaut                  |
//+------------------------------------------------------------------+
void ChangeFiboChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                  double &price2,datetime &time3,double &price3)
  {
//--- si le temps du deuxième point n'est pas spécifié, il sera sur la barre courante
   if(!time2)
      time2=TimeCurrent();
//--- si le prix du deuxième point n'est pas spécifié, il aura la valeur Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du premier (gauche) point n'est pas spécifié, il se trouve au 9 barr
   if(!time1)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //---établissons le premier point au 9 barres de gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas spécifié, rapprocherons-le aux 300 points
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si le temps du troisième point n'est pas spécifié, il coïncide avec le temps du
   if(!time3)
      time3=time1;
//--- si le prix du troisième point n'est pas spécifié, il coïncide avec le prix du de
   if(!price3)
      price3=price2;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- créons"Le canal de Fibonacci"
   if(!FiboChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement du canal
//--- le compteur du cycle
   int h_steps=bars/10;
//--- déplaçons le premier point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d1>1)
         d1-=1;
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- deplaçons le point
      if(!FiboChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   int v_steps=accuracy/10;
//--- déplaçons le deuxième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p2>1)
         p2-=1;
      //--- deplaçons le point
      if(!FiboChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   v_steps=accuracy/15;
//--- déplaçons le troisième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p3<accuracy-1)
         p3+=1;
      //--- deplaçons le point
      if(!FiboChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le canal du graphique
   FiboChannelDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_EXPANSION

L'extension de Fibonacci.

Note

Pour "L'extension de Fibonacci on peut spécifier le mode de son affichage à droite et/ou à gauche (les propriétés OBJPROP_RAY_RIGHT et OBJPROP_RAY_LEFT en conséquence).

On peut aussi spécifier la quantité de lignes-niveau, leurs valeurs et la couleur.

L'exemple

Le script suivant crée et déplace "L'extension de Fibonacci" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"L'extension de Fibonacc
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="FiboExpansion";   // Le nom de l'objet
input int             InpDate1=10;               // La date du 1-ier point en %
input int             InpPrice1=55;              // Le prix du 1-ier point en %
input int             InpDate2=30;               // La date du 2-ième point en %
input int             InpPrice2=10;              // Le prix du 2-ième point en %
input int             InpDate3=80;               // La date du 3-ième point en %
```

© 2000-2015, MetaQuotes Software Corp.

```
input int              InpPrice3=75;              // Le prix du 3-ième point en %
input color            InpColor=clrRed;           // La couleur de l'objet
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Le style des lignes
input int              InpWidth=2;                // L'épaisseur des lignes
input bool             InpBack=false;             // L'objet à l'arrière-plan
input bool             InpSelection=true;         // Sélectionner pour les déplacements
input bool             InpRayLeft=false;          // La suite de l'objet à gauche
input bool             InpRayRight=false;         // La suite de l'objet à droite
input bool             InpHidden=true;            // Est caché dans la liste des objets
input long             InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée "L'extension de Fibonacci" selon les coordonnées spécifiées |
//+------------------------------------------------------------------+
bool FiboExpansionCreate(const long            chart_ID=0,           // ID du graphiqu
                         const string          name="FiboExpansion", // le nom du cana
                         const int             sub_window=0,         // le numéro du s
                         datetime              time1=0,              // le temps du pr
                         double                price1=0,             // le prix du pre
                         datetime              time2=0,              // le temps du de
                         double                price2=0,             // le prix du deu
                         datetime              time3=0,              // le temps du tr
                         double                price3=0,             // le prix du tro
                         const color           clr=clrRed,           // la couleur de
                         const ENUM_LINE_STYLE style=STYLE_SOLID,    // le style des l
                         const int             width=1,              // l'épaisseur de
                         const bool            back=false,           // à l'arrière-pl
                         const bool            selection=true,       // Sélectionner p
                         const bool            ray_left=false,       // la suite de l'
                         const bool            ray_right=false,      // la suite de l'
                         const bool            hidden=true,          // est caché dans
                         const long            z_order=0)            // la priorité au
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeFiboExpansionEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons "L'extension de Fibonacci" selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_EXPANSION,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"L'extension de Fibonacci\"! Le code de l'er
      return(false);
     }
//---  établissons la couleur de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode de la sélection de l'objet pour l
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on r
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage de l'o
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- activons (true) ou désactivons (false) le mode de la suite de l'affichage de l'o
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Spécifie le nombre de niveaux et leurs paramètres                |
//+------------------------------------------------------------------+
bool FiboExpansionLevelsSet(int             levels,               // le nombre de lign
                            double          &values[],            // les valeurs des l
                            color           &colors[],            // la couleur des li
                            ENUM_LINE_STYLE &styles[],            // le style des lign
                            int             &widths[],            // l'épaisseur des l
                            const long      chart_ID=0,           // ID du graphique
                            const string    name="FiboExpansion") // le nom de l'objet
  {
//--- spécifions les tailles des tableaux
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": la longueur du tableau ne correspond pas à la quantité de
      return(false);
     }
//--- établissons la quantité de niveaux
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//---  établissons les propriétés des niveaux dans le cycle
   for(int i=0;i<levels;i++)
     {
      //--- la valeur du niveau
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- la couleur du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- le style du niveau
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- l'épaisseur du niveau
```

© 2000-2015, MetaQuotes Software Corp.

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- la description du niveau
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,"FE "+DoubleToString(100*value
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement "L'extension de Fibonacci"      |
//+------------------------------------------------------------------+
bool FiboExpansionPointChange(const long   chart_ID=0,           // ID du graphique
                              const string name="FiboExpansion", // le nom de l'objet
                              const int    point_index=0,        // le numéro du point
                              datetime     time=0,               // la coordonnée du t
                              double       price=0)              // la coordonnée du p
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime "L'extension de Fibonacci"                              |
//+------------------------------------------------------------------+
bool FiboExpansionDelete(const long   chart_ID=0,           // ID du graphique
                         const string name="FiboExpansion") // le nom de l'objet
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"L'extension de Fibonacci\"! Le code de
      return(false);
     }
//--- l'exécution réussie
```

© 2000-2015, MetaQuotes Software Corp.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement de "L'extension de Fibonacci", et
//| pour les valeurs établit les valeurs par défaut                  |
//+------------------------------------------------------------------+
void ChangeFiboExpansionEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                    double &price2,datetime &time3,double &price3)
  {
//--- si le temps du troisième (droit) point n'est pas spécifié, il sera sur la barre
   if(!time3)
      time3=TimeCurrent();
//--- si le prix du troisième point n'est pas spécifié, il aura la valeur Bid
   if(!price3)
      price3=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du premier (gauche) point n'est pas spécifié, il se trouve au 9 barr
//--- le tableau pour la réception du temps de l'ouverture des 10 dernières barres
   datetime temp[];
   ArrayResize(temp,10);
   if(!time1)
     {
      CopyTime(Symbol(),Period(),time3,10,temp);
      //---établissons le premier point au 9 barres de gauche du deuxième
      time1=temp[0];
     }
//--- si le prix du premier point n'est pas spécifié, il coïncide avec le prix du troi
   if(!price1)
      price1=price3;
//--- si le temps du deuxième point n'est pas spécifié, il se trouve au 7 barres de ga
   if(!time2)
      time2=temp[2];
//--- si le prix du deuxième point n'est pas spécifié, rapprocherons-le aux 250 points
   if(!price2)
      price2=price1-250*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de "L'extension de Fibonacci"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- ccréons "L'extension de Fibonacci"
   if(!FiboExpansionCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement
//--- le compteur du cycle
   int v_steps=accuracy/10;
//--- déplaçons le premier point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1>1)
```

© 2000-2015, MetaQuotes Software Corp.

```
         p1-=1;
       //--- deplaçons le point
       if(!FiboExpansionPointChange(0,InpName,0,date[d1],price[p1]))
          return;
       //--- vérifions le fait de l'arrêt forcé du script
       if(IsStopped())
          return;
       //---redessinons le graphique
       ChartRedraw();
      }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   v_steps=accuracy/2;
//--- déplaçons le troisième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p3>1)
         p3-=1;
      //--- deplaçons le point
      if(!FiboExpansionPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   v_steps=accuracy*4/5;
//--- déplaçons le deuxième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p2<accuracy-1)
         p2+=1;
      //--- deplaçons le point
      if(!FiboExpansionPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
```

© 2000-2015, MetaQuotes Software Corp.

```
   Sleep(1000);
//--- supprimons l'objet du graphique
   FiboExpansionDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ELLIOTWAVE5

La vague d'impulsion d'Elliott

Note

Pour "La vague d'impulsion d'Elliott" on peut activer/désactiver le mode de la liaison des points par les lignes (la propriété OBJPROP_DRAWLINES), ainsi qu'établir le niveau du marquage d'onde (de l'énumération ENUM_ELLIOT_WAVE_DEGREE).

L'exemple

Le script suivant crée et déplace "La vague d'impulsion d'Elliott" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"La vague d'impulsion d'
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string                InpName="ElliotWave5";   // Le nom de l'objet
input int                   InpDate1=10;             // La date du 1-ier point en %
input int                   InpPrice1=90;            // Le prix du 1-ier point en %
input int                   InpDate2=20;             // La date du 2-ième point en %
input int                   InpPrice2=40;            // Le prix du 2-ième point en %
input int                   InpDate3=30;             // La date du 3-ième point en %
```

© 2000-2015, MetaQuotes Software Corp.

```
input int                      InpPrice3=60;            // Le prix du 3-ième point en %
input int                      InpDate4=40;             // La date du 4-ième point en %
input int                      InpPrice4=10;            // Le prix du 4-ième point en %
input int                      InpDate5=60;             // La date du 5-ième point en %
input int                      InpPrice5=40;            // Le prix du 5-ième point en %
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Le niveau
input bool                     InpDrawLines=true;       // L'affichage des lignes
input color                    InpColor=clrRed;         // La couleur des lignes
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // Le style des lignes
input int                      InpWidth=2;              // L'épaisseur des lignes
input bool                     InpBack=false;           // L'objet à l'arrière-plan
input bool                     InpSelection=true;       // Sélectionner pour les déplac
input bool                     InpHidden=true;          // Est caché dans la liste des
input long                     InpZOrder=0;             // La priorité au clic d'une so
//+------------------------------------------------------------------+
//| Crée "La vague d'impulsion d'Elliott" selon les coordonnées spécifiées       |
//+------------------------------------------------------------------+
bool ElliotWave5Create(const long                  chart_ID=0,              // ID du
                       const string                name="ElliotWave5",      // le no
                       const int                   sub_window=0,            // le nu
                       datetime                    time1=0,                 // le te
                       double                      price1=0,                // le pr
                       datetime                    time2=0,                 // le te
                       double                      price2=0,                // le pr
                       datetime                    time3=0,                 // le te
                       double                      price3=0,                // le pr
                       datetime                    time4=0,                 // le te
                       double                      price4=0,                // le pr
                       datetime                    time5=0,                 // le te
                       double                      price5=0,                // le pr
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // le de
                       const bool                  draw_lines=true,         // l'aff
                       const color                 clr=clrRed,              // la co
                       const ENUM_LINE_STYLE       style=STYLE_SOLID,       // le st
                       const int                   width=1,                 // l'épa
                       const bool                  back=false,              // à l'a
                       const bool                  selection=true,          // Sélec
                       const bool                  hidden=true,             // est c
                       const long                  z_order=0)               // la pr
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeElliotWave5EmptyPoints(time1,price1,time2,price2,time3,price3,time4,price4,ti
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons "La vague d'impulsion d'Elliott" selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE5,sub_window,time1,price1,time2,price2
      price3,time4,price4,time5,price5))
     {
      Print(__FUNCTION__,
```

© 2000-2015, MetaQuotes Software Corp.

```
            ": on n'a pas réussi à créer \"La vague d'impulsion d'Elliott\"! Le code d
      return(false);
     }
//--- établissons le degré (la taille de l'onde)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- activons (true) ou désactivons (false) le mode de l'affichage des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//---  établissons la couleur de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode de la sélection de l'objet pour l
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement de "La vague d'impulsion d'Elliott"
//+------------------------------------------------------------------+
bool ElliotWave5PointChange(const long   chart_ID=0,          // ID du graphique
                            const string name="ElliotWave5", // le nom de l'objet
                            const int    point_index=0,      // le numéro du point du
                            datetime     time=0,             // la coordonnée du temps
                            double       price=0)            // la coordonnée du prix
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
```

© 2000-2015, MetaQuotes Software Corp.

```
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime "La vague d'impulsion d'Elliott"                        |
//+------------------------------------------------------------------+
bool ElliotWave5Delete(const long   chart_ID=0,         // ID du graphique
                       const string name="ElliotWave5") // le nom de l'objet
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"La vague d'impulsion d'Elliott\"! Le co
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement de "La vague d'impulsion d'Elliott"
//| pour les valeurs vides établit les valeurs par défaut            |
//+------------------------------------------------------------------+
void ChangeElliotWave5EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3,
                                  datetime &time4,double &price4,
                                  datetime &time5,double &price5)
  {
//--- le tableau pour la réception du temps de l'ouverture des 10 dernières barres
   datetime temp[];
   ArrayResize(temp,10);
//--- recevrons les données
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- recevrons la valeur d'un point sur le graphique courant
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si le temps du premier point n'est pas spécifié, il sera au 9 barres de gauche c
   if(!time1)
      time1=temp[0];
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du deuxième point n'est pas spécifié, il sera au 7 barres de gauche
   if(!time2)
      time2=temp[2];
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- si le prix du deuxième point n'est pas spécifié, rapprocherons-le aux 300 points
   if(!price2)
      price2=price1-300*point;
//--- si le temps du troisième point n'est pas spécifié, il sera au 5 barres de gauche
   if(!time3)
      time3=temp[4];
//--- si le prix du troisième point n'est pas spécifié, déplaçons-le sur 250 points pl
   if(!price3)
      price3=price1-250*point;
//--- si le temps du quatrième point n'est pas spécifié, il sera au 3 barres de gauche
   if(!time4)
      time4=temp[6];
//--- si le prix du quatrième point n'est pas spécifié, déplaçons-le sur 550 points pl
   if(!price4)
      price4=price1-550*point;
//--- si le temps du cinquième point n'est pas spécifié, il sera sur la dernière barre
   if(!time5)
      time5=temp[9];
//--- si le prix du cinquième point n'est pas spécifié, déplaçons-le sur 450 points pl
   if(!price5)
      price5=price1-450*point;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100 ||
      InpDate4<0 || InpDate4>100 || InpPrice4<0 || InpPrice4>100 ||
      InpDate5<0 || InpDate5>100 || InpPrice5<0 || InpPrice5>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
```

© 2000-2015, MetaQuotes Software Corp.

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de "La vague d'impulsion d'Elliott"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int d4=InpDate4*(bars-1)/100;
   int d5=InpDate5*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
   int p4=InpPrice4*(accuracy-1)/100;
   int p5=InpPrice5*(accuracy-1)/100;
//--- créons "La vague d'impulsion d'Elliott"
   if(!ElliotWave5Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      date[d4],price[p4],date[d5],price[p5],InpDegree,InpDrawLines,InpColor,InpStyle,I
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement
//--- le compteur du cycle
   int v_steps=accuracy/5;
//--- déplaçons le cinquième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p5<accuracy-1)
         p5+=1;
      //--- deplaçons le point
      if(!ElliotWave5PointChange(0,InpName,4,date[d5],price[p5]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
```

© 2000-2015, MetaQuotes Software Corp.

```
         return;
       //---redessinons le graphique
       ChartRedraw();
      }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   v_steps=accuracy/5;
//--- déplaçons le deuxième et le troisième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons les valeurs suivantes
      if(p2<accuracy-1)
         p2+=1;
      if(p3>1)
         p3-=1;
      //--- deplaçons les points
      if(!ElliotWave5PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      if(!ElliotWave5PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   v_steps=accuracy*4/5;
//--- déplaçons le premier et le quatrième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons les valeurs suivantes
      if(p1>1)
         p1-=1;
      if(p4<accuracy-1)
         p4+=1;
      //--- deplaçons les points
      if(!ElliotWave5PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave5PointChange(0,InpName,3,date[d4],price[p4]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons l'objet du graphique
   ElliotWave5Delete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ELLIOTWAVE3

La vague de correction d'Elliott.

Note

Pour "la vague de correction d'Elliott"on peut activer/désactiver le mode de la liaison des points par les lignes (la propriétéOBJPROP_DRAWLINES), ainsi qu'établir le niveau du marquage d'onde (de l'énumération ENUM_ELLIOT_WAVE_DEGREE).

L'exemple

Le script suivant crée et déplace "La vague de correction d'Elliott" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"La vague de correction
#property description "Les coordonnées des points du rattachement sont spécifiées en p
#property description "du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string                InpName="ElliotWave3";   // Le nom de l'objet
input int                   InpDate1=10;             // La date du 1-ier point en %
input int                   InpPrice1=90;            // Le prix du 1-ier point en %
input int                   InpDate2=30;             // La date du 2-ième point en %
input int                   InpPrice2=10;            // Le prix du 2-ième point en %
input int                   InpDate3=50;             // La date du 3-ième point en %
```

© 2000-2015, MetaQuotes Software Corp.

```
input int                      InpPrice3=40;            // Le prix du 3-ième point en %
input ENUM_ELLIOT_WAVE_DEGREE  InpDegree=ELLIOTT_MINOR; // Le niveau
input bool                     InpDrawLines=true;       // L'affichage des lignes
input color                    InpColor=clrRed;         // La couleur des lignes
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // Le style des lignes
input int                      InpWidth=2;              // L'épaisseur des lignes
input bool                     InpBack=false;           // L'objet à l'arrière-plan
input bool                     InpSelection=true;       // Sélectionner pour les déplac
input bool                     InpHidden=true;          // Est caché dans la liste des
input long                     InpZOrder=0;             // La priorité au clic d'une so
//+------------------------------------------------------------------+
//| Crée "La vague de correction d'Elliott" selon les coordonnées spécifiées  |
//+------------------------------------------------------------------+
bool ElliotWave3Create(const long                   chart_ID=0,              // ID du
                       const string                 name="ElliotWave3",      // le no
                       const int                    sub_window=0,            // le nu
                       datetime                     time1=0,                 // le te
                       double                       price1=0,                // le pr
                       datetime                     time2=0,                 // le te
                       double                       price2=0,                // le pr
                       datetime                     time3=0,                 // le te
                       double                       price3=0,                // le pr
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // le de
                       const bool                   draw_lines=true,         // l'affi
                       const color                  clr=clrRed,              // la co
                       const ENUM_LINE_STYLE        style=STYLE_SOLID,       // le st
                       const int                    width=1,                 // l'épa
                       const bool                   back=false,              // à l'a
                       const bool                   selection=true,          // Sélec
                       const bool                   hidden=true,             // est c
                       const long                   z_order=0)               // la pr
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeElliotWave3EmptyPoints(time1,price1,time2,price2,time3,price3);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons "La vague de correction d'Elliott" selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE3,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"La vague de correction d'Elliott\"! Le code
      return(false);
     }
//--- établissons le degré (la taille de l'onde)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- activons (true) ou désactivons (false) le mode de l'affichage des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//---  établissons la couleur de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- établissons le style des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode de la sélection de l'objet pour l
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement de "La vague de correction d'Elliott"         |
//+------------------------------------------------------------------+
bool ElliotWave3PointChange(const long   chart_ID=0,          // ID du graphique
                            const string name="ElliotWave3", // le nom de l'objet
                            const int    point_index=0,      // le numéro du point du
                            datetime     time=0,             // la coordonnée du temps
                            double       price=0)            // la coordonnée du prix
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime "La vague de correction d'Elliott"                      |
//+------------------------------------------------------------------+
bool ElliotWave3Delete(const long   chart_ID=0,          // ID du graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
                        const string name="ElliotWave3") // le nom de l'objet
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"La vague de correction d'Elliott\"! Le
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement de "La vague de correction d'Elliot
//| pour les valeurs vides établit les valeurs par défaut            |
//+------------------------------------------------------------------+
void ChangeElliotWave3EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3)
  {
//--- le tableau pour la réception du temps de l'ouverture des 10 dernières barres
   datetime temp[];
   ArrayResize(temp,10);
//--- recevrons les données
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- recevrons la valeur d'un point sur le graphique courant
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si le temps du premier point n'est pas spécifié, il sera au 9 barres de gauche c
   if(!time1)
      time1=temp[0];
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du deuxième point n'est pas spécifié, il sera au 5 barres de gauche
   if(!time2)
      time2=temp[4];
//--- si le prix du deuxième point n'est pas spécifié, rapprocherons-le aux 300 points
   if(!price2)
      price2=price1-300*point;
//--- si le temps du troisième point n'est pas spécifié, il sera à 1 barres de gauche
   if(!time3)
      time3=temp[8];
//--- si le prix du troisième point n'est pas spécifié, déplaçons-le sur 200 points pl
   if(!price3)
      price3=price1-200*point;
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de "La vague de correction d'Elliott"
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- créons "La vague de correction d'Elliott"
   if(!ElliotWave3Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      InpDegree,InpDrawLines,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidder
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement
//--- le compteur du cycle
   int v_steps=accuracy/5;
//--- déplaçons le troisième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p3<accuracy-1)
         p3+=1;
      //--- deplaçons le point
      if(!ElliotWave3PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   v_steps=accuracy*4/5;
//--- déplaçons le premier et le deuxième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons les valeurs suivantes
      if(p1>1)
         p1-=1;
      if(p2<accuracy-1)
         p2+=1;
      //--- deplaçons les points
      if(!ElliotWave3PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave3PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons l'objet du graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
   ElliotWave3Delete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_RECTANGLE

Rectangle.

Note

Pour le rectangle on peut établir aussi le mode du remplissage à l'aide de la propriétéOBJPROP_FILL.

L'exemple

Le script suivant crée et déplace le rectangle sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit le rectangle sur le graphique."
#property description "Les coordonnées des points du rattachement sont spécifiées en"
#property description "pourcentage de la dimension de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="Rectangle"; // Le nom du rectangle
input int             InpDate1=40;         // La date du 1-ier point en %
input int             InpPrice1=40;        // Le prix du 1-ier point en %
input int             InpDate2=60;         // La date du 2-ième point en %
input int             InpPrice2=60;        // Le prix du 2-ième point en %
input color           InpColor=clrRed;     // La couleur du rectangle
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Le style des lignes du rectangle
input int             InpWidth=2;          // L'épaisseur des lignes du rectangle
```

© 2000-2015, MetaQuotes Software Corp.

```
input bool              InpFill=true;         // Le remplissage du rectangle
input bool              InpBack=false;        // Le rectangle à l'arrière-plan
input bool              InpSelection=true;    // Sélectionner pour les déplacements
input bool              InpHidden=true;       // Est caché dans la liste des objets
input long              InpZOrder=0;          // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée le rectangle selon les coordonnées spécifiées               |
//+------------------------------------------------------------------+
bool RectangleCreate(const long            chart_ID=0,        // ID du graphique
                     const string          name="Rectangle",  // le nom du rectangle
                     const int             sub_window=0,      // le numéro du sous-fer
                     datetime              time1=0,           // le temps du premier p
                     double                price1=0,          // le prix du premier po
                     datetime              time2=0,           // le temps du deuxième
                     double                price2=0,          // le prix du deuxième p
                     const color           clr=clrRed,        // la couleur du rectanq
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // le style des lignes d
                     const int             width=1,           // l'épaisseur des ligne
                     const bool            fill=false,        // le remplissage du rec
                     const bool            back=false,        // à l'arrière-plan
                     const bool            selection=true,    // sélectionner pour les
                     const bool            hidden=true,       // est caché dans la lis
                     const long            z_order=0)         // la priorité au clic d
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeRectangleEmptyPoints(time1,price1,time2,price2);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le rectangle selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le rectangle! Le code de l'erreur = ",GetLast
      return(false);
     }
//--- établissons la couleur du rectangle
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style des lignes du rectangle
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes du rectangle
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- activons (true) ou désactivons (false) le mode du remplissage du rectangle
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode de la sélection du rectangle pour
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement du rectangle                    |
//+------------------------------------------------------------------+
bool RectanglePointChange(const long   chart_ID=0,       // ID du graphique
                          const string name="Rectangle", // le nom du rectangle
                          const int    point_index=0,    // le numéro du point du ratt
                          datetime     time=0,           // la coordonnée du temps du
                          double       price=0)          // la coordonnée du prix du p
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le rectangle                                            |
//+------------------------------------------------------------------+
bool RectangleDelete(const long   chart_ID=0,       // ID du graphique
                     const string name="Rectangle") // le nom du rectangle
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprime le rectangle
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le rectangle! Le code de l'erreur = ",Get
      return(false);
```

© 2000-2015, MetaQuotes Software Corp.

```
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement du rectangle, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeRectangleEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- si le temps du premier point n'est pas spécifié, il sera sur la barre courante
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du deuxième point n'est pas spécifié, il se trouve au 9 barres de ga
   if(!time2)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières bari
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- établissons le deuxième point au 9 barres de gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas spécifié, rapprocherons-le aux 300 points
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
```

© 2000-2015, MetaQuotes Software Corp.

```
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du rectangle
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- créons le rectangle
   if(!RectangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement du rectangle
//--- le compteur du cycle
   int h_steps=bars/2;
//--- déplaçons les points du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons les valeurs suivantes
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- deplaçons les points
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   int v_steps=accuracy/2;
//--- déplaçons les points du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons les valeurs suivantes
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- deplaçons les points
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le rectangle
   RectangleDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_TRIANGLE

Triangle.

Note

Pour le triangle on peut établir aussi le mode du remplissage à l'aide de la propriété OBJPROP_FILL.

L'exemple

Le script suivant crée et déplace le triangle sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit le triangle sur le graphique."
#property description "Les coordonnées des points du rattachement sont spécifiées en"
#property description "pourcentage de la dimension de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="Triangle";        // Le nom du triangle
input int             InpDate1=25;               // La date du 1-ier point en %
input int             InpPrice1=50;              // Le prix du 1-ier point en %
input int             InpDate2=70;               // La date du 2-ième point en %
input int             InpPrice2=70;              // Le prix du 2-ième point en %
input int             InpDate3=65;               // La date du 3-ième point en %
input int             InpPrice3=20;              // Le prix du 3-ième point en %
input color           InpColor=clrRed;           // La couleur du triangle
```

© 2000-2015, MetaQuotes Software Corp.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Le style des lignes du triangle
input int             InpWidth=2;                // L'épaisseur des lignes du triangle
input bool            InpFill=false;             // Le remplissage du triangle par la
input bool            InpBack=false;             // Le triangle à l'arrière-plan
input bool            InpSelection=true;         // Sélectionner pour les déplacements
input bool            InpHidden=true;            // Est caché dans la liste des objets
input long            InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée le triangle selon les coordonnées spécifiées                |
//+------------------------------------------------------------------+
bool TriangleCreate(const long            chart_ID=0,        // ID du graphique
                    const string          name="Triangle",   // le nom du triangle
                    const int             sub_window=0,      // le numéro du sous-fené
                    datetime              time1=0,           // le temps du premier po
                    double                price1=0,          // le prix du premier poi
                    datetime              time2=0,           // le temps du deuxième p
                    double                price2=0,          // le prix du deuxième po
                    datetime              time3=0,           // le temps du troisième
                    double                price3=0,          // le prix du troisième p
                    const color           clr=clrRed,        // la couleur du triangle
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // le style des lignes du
                    const int             width=1,           // l'épaisseur des lignes
                    const bool            fill=false,        // le remplissage du tria
                    const bool            back=false,        // à l'arrière-plan
                    const bool            selection=true,    // sélectionner pour les
                    const bool            hidden=true,       // est caché dans la list
                    const long            z_order=0)         // la priorité au clic d'
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeTriangleEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le triangle selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_TRIANGLE,sub_window,time1,price1,time2,price2,ti
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le triangle! Le code de l'erreur = ",GetLastE
      return(false);
     }
//---  établissons la couleur du triangle
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---  établissons le style des lignes du triangle
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes du triangle
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- activons (true) ou désactivons (false) le mode du remplissage du triangle
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- activons (true) ou désactivons (false) le mode de la sélection du triangle pour
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement du triangle                     |
//+------------------------------------------------------------------+
bool TrianglePointChange(const long   chart_ID=0,      // ID du graphique
                         const string name="Triangle", // le nom du triangle
                         const int    point_index=0,   // le numéro du point du rattac
                         datetime     time=0,          // la coordonnée du temps du po
                         double       price=0)         // la coordonnée du prix du poi
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le triangle                                             |
//+------------------------------------------------------------------+
bool TriangleDelete(const long   chart_ID=0,      // ID du graphique
                    const string name="Triangle") // le nom du triangle
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le triangle
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le triangle! Le code de l'erreur = ",GetI
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement du triangle, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeTriangleEmptyPoints(datetime &time1,double &price1,
                               datetime &time2,double &price2,
                               datetime &time3,double &price3)
  {
//--- si le temps du premier point n'est pas spécifié, il sera sur la barre courante
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du deuxième point n'est pas spécifié, il se trouve au 9 barres de ga
   if(!time2)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- établissons le deuxième point au 9 barres de gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas spécifié, rapprocherons-le aux 300 points
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si le temps du troisième point n'est pas spécifié, il coïncide avec la date du c
   if(!time3)
      time3=time2;
//--- si le prix du premier point n'est pas spécifié, il coïncide avec le prix du prem
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
```

© 2000-2015, MetaQuotes Software Corp.

```
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du triangle
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- créons le triangle
   if(!TriangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement du triangle
//--- le compteur du cycle
```

© 2000-2015, MetaQuotes Software Corp.

```
   int v_steps=accuracy*3/10;
//--- déplaçons le premier point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p1>1)
         p1-=1;
      //--- deplaçons le point
      if(!TrianglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   int h_steps=bars*9/20-1;
//--- déplaçons le deuxième point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d2>1)
         d2-=1;
      //--- deplaçons le point
      if(!TrianglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   v_steps=accuracy/4;
//--- déplaçons le troisième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p3<accuracy-1)
         p3+=1;
      //--- deplaçons le point
      if(!TrianglePointChange(0,InpName,2,date[d3],price[p3]))
```

© 2000-2015, MetaQuotes Software Corp.

```
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le triangle du graphique
   TriangleDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ELLIPSE

Ellipse.

Note

Pour l'ellipse on peut établir aussi le mode du remplissage à l'aide de la propriété OBJPROP_FILL.

L'exemple

Le script suivant crée et déplace l'ellipse sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'ellipse sur le graphique."
#property description "Les coordonnées des points du rattachement sont spécifiées en"
#property description "pourcentage de la dimension de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpName="Ellipse";         // Le nom de l'ellipse
input int             InpDate1=30;               // La date du 1-ier point en %
input int             InpPrice1=20;              // Le prix du 1-ier point en %
input int             InpDate2=70;               // La date du 2-ième point en %
input int             InpPrice2=80;              // Le prix du 2-ième point en %
input int             InpDate3=50;               // La date du 3-ième point en %
input int             InpPrice3=60;              // La date du 3-ième point en %
input color           InpColor=clrRed;           // La couleur de l'ellipse
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Le style des lignes de l'ellipse
```

© 2000-2015, MetaQuotes Software Corp.

```
input int              InpWidth=2;                // L'épaisseur des lignes de l'ellips
input bool             InpFill=false;             // Le remplissage de l'ellipse par la
input bool             InpBack=false;             // l'ellipse à l'arrière-plan
input bool             InpSelection=true;         // Sélectionner pour les déplacements
input bool             InpHidden=true;            // Est caché dans la liste des objets
input long             InpZOrder=0;               // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée l'ellipse selon les coordonnées spécifiées                  |
//+------------------------------------------------------------------+
bool EllipseCreate(const long            chart_ID=0,        // ID du graphique
                   const string          name="Ellipse",    // le nom de l'ellipse
                   const int             sub_window=0,      // le numéro du sous-fenêt
                   datetime              time1=0,           // le temps du premier poi
                   double                price1=0,          // le prix du premier poin
                   datetime              time2=0,           // le temps du deuxième po
                   double                price2=0,          // le prix du deuxième poi
                   datetime              time3=0,           // le temps du troisième p
                   double                price3=0,          // le prix du troisième po
                   const color           clr=clrRed,        // la couleur de l'ellipse
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // le style des lignes de
                   const int             width=1,           // l'épaisseur des lignes
                   const bool            fill=false,        // le remplissage de l'ell
                   const bool            back=false,        // à l'arrière-plan
                   const bool            selection=true,    // sélectionner pour les d
                   const bool            hidden=true,       // est caché dans la liste
                   const long            z_order=0)         // la priorité au clic d'u
  {
//---établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeEllipseEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons l'ellipse selon les coordonnées spécifiées
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIPSE,sub_window,time1,price1,time2,price2,tim
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer l'ellipse! Le code de l'erreur = ",GetLastErr
      return(false);
     }
//--- établissons la couleur de l'ellipse
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style des lignes de l'ellipse
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur des lignes de l'ellipse
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- activons (true) ou désactivons (false) le mode du remplissage de l'ellipse
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode de la sélection de l'ellipse pour
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on r
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement de l'ellipse                    |
//+------------------------------------------------------------------+
bool EllipsePointChange(const long   chart_ID=0,     // ID du graphique
                        const string name="Ellipse", // le nom de l'ellipse
                        const int    point_index=0,  // le numéro du point du rattache
                        datetime     time=0,         // la coordonnée du temps du poin
                        double       price=0)        // la coordonnée du prix du point
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime l'ellipse                                               |
//+------------------------------------------------------------------+
bool EllipseDelete(const long   chart_ID=0,     // ID du graphique
                   const string name="Ellipse") // le nom de l'ellipse
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons l'ellipse
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer l'ellipse! Le code de l'erreur = ",GetLas
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs des points du rattachement de la ligne, et pour les valeurs |
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeEllipseEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2,
                              datetime &time3,double &price3)
  {
//--- si le temps du premier point n'est pas spécifié, il sera sur la barre courante
   if(!time1)
      time1=TimeCurrent();
//--- si le prix du premier point n'est pas spécifié, il aura la valeur Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- si le temps du deuxième point n'est pas spécifié, il se trouve au 9 barres de ga
   if(!time2)
     {
      //--- le tableau pour la réception du temps de l'ouverture des 10 dernières barr
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- établissons le deuxième point au 9 barres de gauche du premier
      time2=temp[0];
     }
//--- si le prix du deuxième point n'est pas spécifié, rapprocherons-le aux 300 points
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- si le temps du troisième point n'est pas spécifié, il coïncide avec la date du c
   if(!time3)
      time3=time2;
//--- si le prix du troisième point n'est pas spécifié, il coïncide avec le prix du pr
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de l'ellipse
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- créons l'ellipse
   if(!EllipseCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons les points du rattachement de l'ellipse
//--- le compteur du cycle
   int v_steps=accuracy/5;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- déplaçons le premier et le deuxième point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons les valeurs suivantes
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- deplaçons les points
      if(!EllipsePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!EllipsePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   int h_steps=bars/5;
//--- déplaçons le troisième point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d3>1)
         d3-=1;
      //--- deplaçons le point
      if(!EllipsePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons l'ellipse du graphique
   EllipseDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW_THUMB_UP

Le signe "le Doigt en haut".

Note

La position du point du rattachement par rapport le signe peut être choisie de l'énumération ENUM_ARROW_ANCHOR.

On peut créer les signes d'une grande taille (plus que 5) ayant établi seulement la valeur correspondante de la propriété OBJPROP_WIDTHà l'écriture du code dans MetaEditor.

L'exemple

Le script suivant crée et déplace le signe "le Doigt en haut" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script dessine le signe \"le Doigt en haut\".
#property description "La coordonnée du point du rattachement est spécifiée en pourcer
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string             InpName="ThumbUp";     // Le nom du signe
input int                InpDate=75;            // La date du point du rattachement en %
input int                InpPrice=25;           // Le prix du point du rattachement en %
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;  // Le moyen du rattachement
input color              InpColor=clrRed;       // La couleur du signe
```

© 2000-2015, MetaQuotes Software Corp.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Le style de la ligne bordant
input int               InpWidth=5;           // La taille du signe
input bool              InpBack=false;        // Le signe à l'arrière-plan
input bool              InpSelection=true;    // Sélectionner pour les déplacements
input bool              InpHidden=true;       // Est caché dans la liste des objets
input long              InpZOrder=0;          // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée le signe "le Doigt en haut"                                 |
//+------------------------------------------------------------------+
bool ArrowThumbUpCreate(const long              chart_ID=0,           // ID du graphic
                        const string            name="ThumbUp",       // le nom du sig
                        const int               sub_window=0,         // le numéro du
                        datetime                time=0,               // le temps du p
                        double                  price=0,              // le prix du po
                        const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // le moyen du r
                        const color             clr=clrRed,           // la couleur du
                        const ENUM_LINE_STYLE   style=STYLE_SOLID,    // le style de l
                        const int               width=3,              // la taille du
                        const bool              back=false,           // à l'arrière-p
                        const bool              selection=true,       // Sélectionner
                        const bool              hidden=true,          // est caché dan
                        const long              z_order=0)            // la priorité a
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécif
   ChangeArrowEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le signe \"le Doigt en haut\"! Le code de l'e
      return(false);
     }
//--- établissons le moyen du rattachement
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- établissons la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne bordant
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement du signe par la so
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on r
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
bool ArrowThumbUpMove(const long   chart_ID=0,      // ID du graphique
                      const string name="ThumbUp", // le nom de l'objet
                      datetime     time=0,         // la coordonnée du temps du point
                      double       price=0)        // la coordonnée du prix du point
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le moyen du rattachement du signe "le Doigt en haut"
//+------------------------------------------------------------------+
bool ArrowThumbUpAnchorChange(const long              chart_ID=0,        // ID du grap
                              const string            name="ThumbUp",    // le nom de
                              const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // le moyen
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- chageons le moyen du rattachement
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le moyen du rattachement! Le code de l'erre
      return(false);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le signe "le Doigt en haut"                             |
//+------------------------------------------------------------------+
bool ArrowThumbUpDelete(const long   chart_ID=0,     // ID du graphique
                        const string name="ThumbUp") // le nom du signe
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le signe\"le Doigt en haut\"! Le code de
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- créons le signe "le Doigt en haut" sur le graphique.
   if(!ArrowThumbUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//---maintenant déplaçons le point du rattachement et changeons son position par rappo
//--- le compteur du cycle
   int h_steps=bars/4;
//--- déplaçons le point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d>1)
         d-=1;
      //--- deplaçons le point
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
```

© 2000-2015, MetaQuotes Software Corp.

```
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   int v_steps=accuracy/4;
//--- déplaçons le point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p<accuracy-1)
         p+=1;
      //--- deplaçons le point
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- changeons la position du point du rattachement par rapport au signe
   ArrowThumbUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redessinons le graphique
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le signe du graphique
   ArrowThumbUpDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW_THUMB_DOWN

Le signe "le Doigt en bas".

Note

La position du point du rattachement par rapport le signe peut être choisie de l'énumération ENUM_ARROW_ANCHOR.

On peut créer les signes d'une grande taille (plus que 5) ayant établi seulement la valeur correspondante de la propriété OBJPROP_WIDTHà l'écriture du code dans MetaEditor.

L'exemple

Le script suivant crée et déplace le signe "le Doigt en bas" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script dessine le signe \""le Doigt en bas"\.
#property description "La coordonnée du point du rattachement est spécifiée en pourcer
#property description "la taille de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string            InpName="ThumbDown";      // Le nom du signe
input int               InpDate=25;               // La date du point du rattachement e
input int               InpPrice=75;              // Le prix du point du rattachement e
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // Le moyen du rattachement
```

© 2000-2015, MetaQuotes Software Corp.

```
input color             InpColor=clrRed;         // La couleur du signe
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Le style de la ligne bordant
input int               InpWidth=5;              // La taille du signe
input bool              InpBack=false;           // Le signe à l'arrière-plan
input bool              InpSelection=true;       // Sélectionner pour les déplacements
input bool              InpHidden=true;          // Est caché dans la liste des objets
input long              InpZOrder=0;             // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée le signe "le Doigt en bas"                                  |
//+------------------------------------------------------------------+
bool ArrowThumbDownCreate(const long              chart_ID=0,           // ID du graph
                          const string            name="ThumbDown",     // le nom du s
                          const int               sub_window=0,         // le numéro d
                          datetime                time=0,               // le temps du
                          double                  price=0,              // le prix du
                          const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // le moyen du
                          const color             clr=clrRed,           // la couleur
                          const ENUM_LINE_STYLE   style=STYLE_SOLID,    // le style de
                          const int               width=3,              // la taille d
                          const bool              back=false,           // à l'arrière
                          const bool              selection=true,       // Sélectionne
                          const bool              hidden=true,          // est caché d
                          const long              z_order=0)            // la priorité
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spéci
   ChangeArrowEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le signe \"le Doigt en bas\"! Le code de l'er
      return(false);
     }
//--- établissons le moyen du rattachement
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- établissons la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne bordant
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement du signe par la so
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
bool ArrowThumbDownMove(const long   chart_ID=0,       // ID du graphique
                        const string name="ThumbDown", // le nom de l'objet
                        datetime     time=0,           // la coordonnée du temps du po
                        double       price=0)          // la coordonnée du prix du poi
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le moyen du rattachement du signe "le Doigt en bas"
//+------------------------------------------------------------------+
bool ArrowThumbDownAnchorChange(const long              chart_ID=0,        // ID du gr
                                const string            name="ThumbDown",  // le nom d
                                const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // le moyen
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- chageons le moyen du rattachement
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le moyen du rattachement! Le code de l'erre
      return(false);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le signe "le Doigt en bas"                              |
//+------------------------------------------------------------------+
bool ArrowThumbDownDelete(const long   chart_ID=0,       // ID du graphique
                          const string name="ThumbDown") // le nom du signe
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le signe \"le Doigt en bas\"! Le code de
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
```

© 2000-2015, MetaQuotes Software Corp.

```
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//---  créons le signe "le Doigt en bas" sur le graphique.
   if(!ArrowThumbDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//---maintenant déplaçons le point du rattachement et changeons son position par rappo
//--- le compteur du cycle
   int h_steps=bars/4;
//--- déplaçons le point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- deplaçons le point
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
```

© 2000-2015, MetaQuotes Software Corp.

```
         return;
       //---redessinons le graphique
       ChartRedraw();
       // le retard à 0.05 seconde
       Sleep(50);
      }
//--- le retard à 1 seconde
   Sleep(1000);
//--- le compteur du cycle
   int v_steps=accuracy/4;
//--- déplaçons le point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p>1)
         p-=1;
      //--- deplaçons le point
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- changeons la position du point du rattachement par rapport au signe
   ArrowThumbDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- redessinons le graphique
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le signe du graphique
   ArrowThumbDownDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW_UP

Le signe "La flèche en haut".

Note

La position du point du rattachement par rapport le signe peut être choisie de l'énumération ENUM_ARROW_ANCHOR.

On peut créer les signes d'une grande taille (plus que 5) ayant établi seulement la valeur correspondante de la propriété OBJPROP_WIDTHà l'écriture du code dans MetaEditor.

L'exemple

Le script suivant crée et déplace le signe "La flèche en haut" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script dessine le signe \"La flèche en haut\"."
#property description "La coordonnée du point du rattachement est spécifiée en"
#property description "pourcentage de la dimension de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string            InpName="ArrowUp";     // Le nom du signe
input int               InpDate=25;            // La date du point du rattachement en %
input int               InpPrice=25;           // Le prix du point du rattachement en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // Le moyen du rattachement
input color             InpColor=clrRed;       // La couleur du signe
```

© 2000-2015, MetaQuotes Software Corp.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Le style de la ligne bordant
input int               InpWidth=5;           // La taille du signe
input bool              InpBack=false;        // Le signe à l'arrière-plan
input bool              InpSelection=false;   // Sélectionner pour les déplacements
input bool              InpHidden=true;       // Est caché dans la liste des objets
input long              InpZOrder=0;          // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée le signe "La flèche en haut"                                |
//+------------------------------------------------------------------+
bool ArrowUpCreate(const long              chart_ID=0,           // ID du graphique
                   const string            name="ArrowUp",       // le nom du signe
                   const int               sub_window=0,         // le numéro du sous-
                   datetime                time=0,               // le temps du point
                   double                  price=0,              // le prix du point d
                   const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // le moyen du rattac
                   const color             clr=clrRed,           // la couleur du sign
                   const ENUM_LINE_STYLE   style=STYLE_SOLID,    // le style de la lig
                   const int               width=3,              // la taille du signe
                   const bool              back=false,           // à l'arrière-plan
                   const bool              selection=true,       // Sélectionner pour
                   const bool              hidden=true,          // est caché dans la
                   const long              z_order=0)            // la priorité au cli
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécif
   ChangeArrowEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le signe \""La flèche en haut"\"! Le code de
      return(false);
     }
//--- établissons le moyen du rattachement
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- établissons la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne bordant
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement du signe par la so
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
bool ArrowUpMove(const long   chart_ID=0,     // ID du graphique
                 const string name="ArrowUp", // le nom de l'objet
                 datetime     time=0,         // la coordonnée du temps du point du ra
                 double       price=0)        // la coordonnée du prix du point du rat
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le moyen du rattachement du signe "La flèche en haut"     |
//+------------------------------------------------------------------+
bool ArrowUpAnchorChange(const long              chart_ID=0,        // ID du graphique
                         const string            name="ArrowUp",    // le nom de l'obj
                         const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // le moyen du rat
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons la position du point du rattachement
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le moyen du rattachement! Le code de l'erre
      return(false);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le signe "La flèche en haut"                            |
//+------------------------------------------------------------------+
bool ArrowUpDelete(const long   chart_ID=0,     // ID du graphique
                   const string name="ArrowUp") // le nom du signe
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le signe \"Le signe "La flèche en haut"\"
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- créons le signe "La flèche en haut" sur le graphique
   if(!ArrowUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//---maintenant déplaçons le point du rattachement et changeons son position par rappo
//--- le compteur du cycle
   int v_steps=accuracy/2;
//--- déplaçons le point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p<accuracy-1)
         p+=1;
      //--- deplaçons le point
      if(!ArrowUpMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
```

© 2000-2015, MetaQuotes Software Corp.

```
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- changeons la position du point du rattachement par rapport au signe
   ArrowUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redessinons le graphique
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le signe du graphique
   ArrowUpDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW_DOWN

Le signe "La flèche en bas".

Note

La position du point du rattachement par rapport le signe peut être choisie de l'énumération ENUM_ARROW_ANCHOR.

On peut créer les signes d'une grande taille (plus que 5) ayant établi seulement la valeur correspondante de la propriété OBJPROP_WIDTHà l'écriture du code dans MetaEditor.

L'exemple

Le script suivant crée et déplace le signe "La flèche en bas" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script dessine le signe \"La flèche en bas\"."
#property description "La coordonnée du point du rattachement est spécifiée en"
#property description "pourcentage de la dimension de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string            InpName="ArrowDown";      // Le nom du signe
input int               InpDate=75;               // La date du point du rattachement e
input int               InpPrice=75;              // Le prix du point du rattachement e
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // Le moyen du rattachement
input color             InpColor=clrRed;          // La couleur du signe
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;       // Le style de la ligne bordant
```

© 2000-2015, MetaQuotes Software Corp.

```
input int                InpWidth=5;              // La taille du signe
input bool               InpBack=false;           // Le signe à l'arrière-plan
input bool               InpSelection=false;      // Sélectionner pour les déplacements
input bool               InpHidden=true;          // Est caché dans la liste des objets
input long               InpZOrder=0;             // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée le signe "La flèche en bas"                                 |
//+------------------------------------------------------------------+
bool ArrowDownCreate(const long              chart_ID=0,           // ID du graphique
                     const string            name="ArrowDown",     // le nom du signe
                     const int               sub_window=0,         // le numéro du sou
                     datetime                time=0,               // le temps du poin
                     double                  price=0,              // le prix du point
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // le moyen du ratt
                     const color             clr=clrRed,           // la couleur du si
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // le style de la l
                     const int               width=3,              // la taille du sig
                     const bool              back=false,           // à l'arrière-plan
                     const bool              selection=true,       // Sélectionner pou
                     const bool              hidden=true,          // est caché dans l
                     const long              z_order=0)            // la priorité au c
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeArrowEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"La flèche en bas\"! Le code de l'erreur = "
      return(false);
     }
//--- le moyen du rattachement
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- établissons la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne bordant
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement du signe par la so
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
bool ArrowDownMove(const long   chart_ID=0,       // ID du graphique
                   const string name="ArrowDown", // le nom de l'objet
                   datetime     time=0,           // la coordonnée du temps du point d
                   double       price=0)          // la coordonnée du prix du point du
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le moyen du rattachement du signe "La flèche en bas"      |
//+------------------------------------------------------------------+
bool ArrowDownAnchorChange(const long              chart_ID=0,        // ID du graphic
                           const string            name="ArrowDown",  // le nom de l'o
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // le moyen du r
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons la position du point du rattachement
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le moyen du rattachement! Le code de l'erre
      return(false);
     }
//--- l'exécution réussie
```

© 2000-2015, MetaQuotes Software Corp.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le signe "La flèche en bas"                             |
//+------------------------------------------------------------------+
bool ArrowDownDelete(const long   chart_ID=0,       // ID du graphique
                     const string name="ArrowDown") // le nom du signe
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le signe\"La flèche en bas\"! Le code de
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- créons le signe "La flèche en bas" sur le graphique
   if(!ArrowDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//---maintenant déplaçons le point du rattachement et changeons son position par rappo
//--- le compteur du cycle
   int v_steps=accuracy/2;
//--- déplaçons le point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p>1)
         p-=1;
      //--- deplaçons le point
      if(!ArrowDownMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- changeons la position du point du rattachement par rapport au signe
   ArrowDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- redessinons le graphique
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le signe du graphique
   ArrowDownDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW_STOP

Le signe "Stop".

Note

La position du point du rattachement par rapport le signe peut être choisie de l'énumération ENUM_ARROW_ANCHOR.

On peut créer les signes d'une grande taille (plus que 5) ayant établi seulement la valeur correspondante de la propriété OBJPROP_WIDTHà l'écriture du code dans MetaEditor.

L'exemple

Le script suivant crée et déplace le signe "Stop" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script dessine le signe \"Stop\"."
#property description "La coordonnée du point du rattachement est spécifiée en"
#property description "pourcentage de la dimension de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string             InpName="ArrowStop";      // Le nom du signe
input int                InpDate=10;               // La date du point du rattachement e
input int                InpPrice=50;              // Le prix du point du rattachement e
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_BOTTOM;  // Le moyen du rattachement
input color              InpColor=clrRed;          // La couleur du signe
```

© 2000-2015, MetaQuotes Software Corp.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Le style de la ligne bordant
input int               InpWidth=5;              // La taille du signe
input bool              InpBack=false;           // Le signe à l'arrière-plan
input bool              InpSelection=false;      // Sélectionner pour les déplacements
input bool              InpHidden=true;          // Est caché dans la liste des objets
input long              InpZOrder=0;             // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée le signe "Stop"                                             |
//+------------------------------------------------------------------+
bool ArrowStopCreate(const long              chart_ID=0,           // ID du graphique
                     const string            name="ArrowStop",     // le nom du signe
                     const int               sub_window=0,         // le numéro du sou
                     datetime                time=0,               // le temps du poin
                     double                  price=0,              // le prix du point
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // le moyen du ratt
                     const color             clr=clrRed,           // la couleur du si
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // le style de la l
                     const int               width=3,              // la taille du sig
                     const bool              back=false,           // à l'arrière-plan
                     const bool              selection=true,       // Sélectionner pou
                     const bool              hidden=true,          // est caché dans l
                     const long              z_order=0)            // la priorité au c
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécif
   ChangeArrowEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_STOP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le signe \"Stop\"! Le code de l'erreur = ",Ge
      return(false);
     }
//--- établissons le moyen du rattachement
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- établissons la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne bordant
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement du signe par la so
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
bool ArrowStopMove(const long   chart_ID=0,       // ID du graphique
                   const string name="ArrowStop", // le nom de l'objet
                   datetime     time=0,           // la coordonnée du temps du point d
                   double       price=0)          // la coordonnée du prix du point du
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le moyen du rattachement du signe "Stop"                  |
//+------------------------------------------------------------------+
bool ArrowStopAnchorChange(const long              chart_ID=0,        // ID du graphic
                           const string            name="ArrowStop",  // le nom de l'o
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // la position d
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- chageons le moyen du rattachement
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le moyen du rattachement! Le code de l'erre
      return(false);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le signe "Stop"                                         |
//+------------------------------------------------------------------+
bool ArrowStopDelete(const long   chart_ID=0,       // ID du graphique
                     const string name="ArrowStop") // le nom du signe
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le signe \"Stop\"! Le code de l'erreur =
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- créons le signe "Stop" sur le graphique
   if(!ArrowStopCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//---maintenant déplaçons le point du rattachement et changeons son position par rappo
//--- le compteur du cycle
   int h_steps=bars*2/5;
//--- déplaçons le point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- deplaçons le point
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
```

© 2000-2015, MetaQuotes Software Corp.

```
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.025 secondes
      Sleep(25);
     }
//--- changeons la position du point du rattachement par rapport au signe
   ArrowStopAnchorChange(0,InpName,ANCHOR_TOP);
//--- redessinons le graphique
   ChartRedraw();
//--- le compteur du cycle
   h_steps=bars*2/5;
//--- déplaçons le point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- deplaçons le point
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.025 secondes
      Sleep(25);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le signe du graphique
   ArrowStopDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW_CHECK

Le signe "Coche".

Note

La position du point du rattachement par rapport le signe peut être choisie de l'énumération ENUM_ARROW_ANCHOR.

On peut créer les signes d'une grande taille (plus que 5) ayant établi seulement la valeur correspondante de la propriété OBJPROP_WIDTHà l'écriture du code dans MetaEditor.

L'exemple

Le script suivant crée et déplace le signe "Coche" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script dessine le signe \"Coche\"."
#property description "La coordonnée du point du rattachement est spécifiée en"
#property description "pourcentage de la dimension de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string            InpName="ArrowCheck"; // Le nom du signe
input int               InpDate=10;           // La date du point du rattachement en %
input int               InpPrice=50;          // Le prix du point du rattachement en %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // Le moyen du rattachement
input color             InpColor=clrRed;      // La couleur du signe
```

© 2000-2015, MetaQuotes Software Corp.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Le style de la ligne bordant
input int               InpWidth=5;           // La taille du signe
input bool              InpBack=false;        // Le signe à l'arrière-plan
input bool              InpSelection=false;   // Sélectionner pour les déplacements
input bool              InpHidden=true;       // Est caché dans la liste des objets
input long              InpZOrder=0;          // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée le signe "Coche"                                            |
//+------------------------------------------------------------------+
bool ArrowCheckCreate(const long              chart_ID=0,           // ID du graphique
                      const string            name="ArrowCheck",    // le nom du signe
                      const int               sub_window=0,         // le numéro du sc
                      datetime                time=0,               // le temps du poi
                      double                  price=0,              // le prix du poin
                      const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // le moyen du rat
                      const color             clr=clrRed,           // la couleur du s
                      const ENUM_LINE_STYLE   style=STYLE_SOLID,    // le style de la
                      const int               width=3,              // la taille du si
                      const bool              back=false,           // à l'arrière-pla
                      const bool              selection=true,       // Sélectionner po
                      const bool              hidden=true,          // est caché dans
                      const long              z_order=0)            // la priorité au
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécif
   ChangeArrowEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_CHECK,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"Coche\"! Le code de l'erreur = ",GetLastErr
      return(false);
     }
//--- établissons le moyen du rattachement
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- établissons la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne bordant
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons la taille du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement du signe par la so
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
bool ArrowCheckMove(const long   chart_ID=0,        // ID du graphique
                    const string name="ArrowCheck", // le nom de l'objet
                    datetime     time=0,            // la coordonnée du temps du point
                    double       price=0)           // la coordonnée du prix du point
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le moyen du rattachement de "la Coche"                    |
//+------------------------------------------------------------------+
bool ArrowCheckAnchorChange(const long              chart_ID=0,        // ID du graphi
                            const string            name="ArrowCheck", // le nom de l'
                            const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // le moyen du
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- chageons le moyen du rattachement
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le moyen du rattachement! Le code de l'erre
      return(false);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le signe "Coche"                                        |
//+------------------------------------------------------------------+
bool ArrowCheckDelete(const long   chart_ID=0,        // ID du graphique
                      const string name="ArrowCheck") // le nom du signe
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le signe \"Coche\"! Le code de l'erreur =
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin du signe
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- créons le signe "Coche" sur le graphique
   if(!ArrowCheckCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//---maintenant déplaçons le point du rattachement et changeons son position par rappo
//--- le compteur du cycle
   int h_steps=bars*2/5;
//--- déplaçons le point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- deplaçons le point
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
```

© 2000-2015, MetaQuotes Software Corp.

```
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.025 secondes
      Sleep(25);
     }
//--- changeons la position du point du rattachement par rapport au signe
   ArrowCheckAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redessinons le graphique
   ChartRedraw();
//--- le compteur du cycle
   h_steps=bars*2/5;
//--- déplaçons le point du rattachement
   for(int i=0;i<h_steps;i++)
     {
      //--- prenons la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- deplaçons le point
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.025 secondes
      Sleep(25);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le signe du graphique
   ArrowCheckDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW_LEFT_PRICE

La marque gauche de prix.

L'exemple

Le script suivant crée et déplace la marque gauche de prix sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script crée la marque gauche de prix sur le graphique."
#property description "La coordonnée du point du rattachement est spécifiée en pourcer
#property description "de la dimension de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string            InpName="LeftPrice";  // Le nom de la marque de prix
input int               InpDate=100;          // La date du point du rattachement en %
input int               InpPrice=10;          // Le prix du point du rattachement en %
input color             InpColor=clrRed;      // La couleur de la marque droite de pri
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Le style de la ligne bordant
input int               InpWidth=2;           // La taille de la marque de prix
input bool              InpBack=false;        // La marque à l'arrière-plan
input bool              InpSelection=true;    // Sélectionner pour les déplacements
input bool              InpHidden=true;       // Est caché dans la liste des objets
input long              InpZOrder=0;          // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée la marque gauche de prix                                    |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
bool ArrowLeftPriceCreate(const long            chart_ID=0,        // ID du graphique
                          const string          name="LeftPrice",  // le nom de la mar
                          const int             sub_window=0,      // le numéro du sou
                          datetime              time=0,            // le temps du poin
                          double                price=0,           // le prix du point
                          const color           clr=clrRed,        // la couleur de la
                          const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la l
                          const int             width=1,           // la taille de la
                          const bool            back=false,        // à l'arrière-plan
                          const bool            selection=true,    // Sélectionner pou
                          const bool            hidden=true,       // est caché dans l
                          const long            z_order=0)         // la priorité au c
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécif
   ChangeArrowEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons la marque de prix
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_LEFT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer la marque gauche de prix! Le code de l'erreur
      return(false);
     }
//--- établissons la couleur de la marque
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne bordant
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//---  établissons la taille de la marque
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement de la marque par l
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool ArrowLeftPriceMove(const long   chart_ID=0,       // ID du graphique
                        const string name="LeftPrice", // le nom de la marque
                        datetime     time=0,           // la coordonnée du temps du po
                        double       price=0)          // la coordonnée du prix du poi
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime la marque gauche de prix du graphique                   |
//+------------------------------------------------------------------+
bool ArrowLeftPriceDelete(const long   chart_ID=0,       // ID du graphique
                          const string name="LeftPrice") // le nom de la marque
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la marque
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer la marque gauche de prix! Le code de l'er
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de la marque
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- créons la marque gauche de prix sur le graphique
   if(!ArrowLeftPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons le point du rattachement
//--- le compteur du cycle
   int v_steps=accuracy*4/5;
//--- déplaçons le point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p<accuracy-1)
         p+=1;
      //--- deplaçons le point
      if(!ArrowLeftPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons la marque du graphique
   ArrowLeftPriceDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW_RIGHT_PRICE

La marque droite de prix.

L'exemple

Le script suivant crée et déplace la marque droite de prix sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script crée la marque droite de prix sur le graphique."
#property description "La coordonnée du point du rattachement est spécifiée en pourcer
#property description "de la dimension de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string            InpName="RightPrice"; // Le nom de la marque droite de prix
input int               InpDate=0;            // La date du point du rattachement en %
input int               InpPrice=90;          // Le prix du point du rattachement en %
input color             InpColor=clrRed;      // La couleur de la marque droite de pri
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Le style de la ligne bordant
input int               InpWidth=2;           // La taille de la marque de prix
input bool              InpBack=false;        // La marque à l'arrière-plan
input bool              InpSelection=true;    // Sélectionner pour les déplacements
input bool              InpHidden=true;       // Est caché dans la liste des objets
input long              InpZOrder=0;          // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée la marque droite de prix                                    |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
bool ArrowRightPriceCreate(const long            chart_ID=0,        // ID du graphique
                           const string          name="RightPrice", // le nom de la ma
                           const int             sub_window=0,      // le numéro du so
                           datetime              time=0,            // le temps du poi
                           double                price=0,           // le prix du poin
                           const color           clr=clrRed,        // la couleur de l
                           const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la
                           const int             width=1,           // la taille de la
                           const bool            back=false,        // à l'arrière-pla
                           const bool            selection=true,    // Sélectionner po
                           const bool            hidden=true,       // est caché dans
                           const long            z_order=0)         // la priorité au
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécif
   ChangeArrowEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons la marque de prix
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_RIGHT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer la marque droite de prix! Le code de l'erreur
      return(false);
     }
//--- établissons la couleur de la marque
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne bordant
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//---  établissons la taille de la marque
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement de la marque par l
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool ArrowRightPriceMove(const long   chart_ID=0,        // ID du graphique
                         const string name="RightPrice", // le nom de la marque
                         datetime     time=0,            // la coordonnée du temps du
                         double       price=0)           // la coordonnée du prix du p
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime la marque droite de prix du graphique                   |
//+------------------------------------------------------------------+
bool ArrowRightPriceDelete(const long   chart_ID=0,        // ID du graphique
                           const string name="RightPrice") // le nom de la marque
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la marque
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer la marque droite de prix! Le code de l'er
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de la marque
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- créons la marque droite de prix sur le graphique
   if(!ArrowRightPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

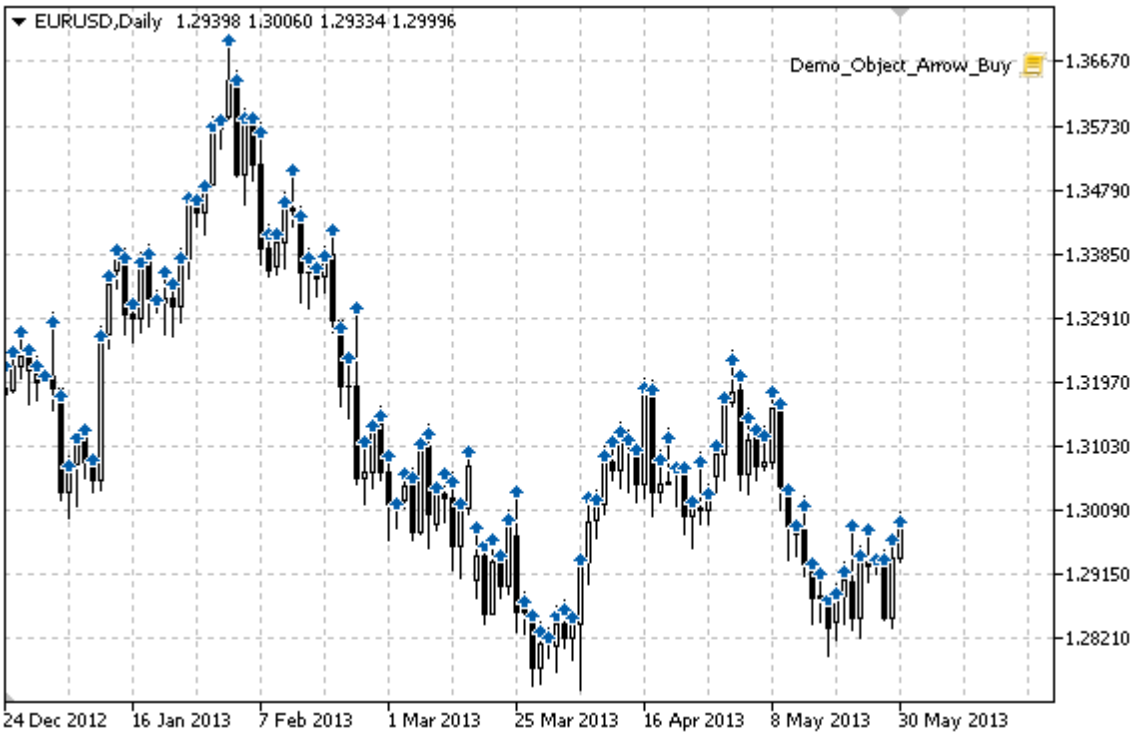
© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons le point du rattachement
//--- le compteur du cycle
   int v_steps=accuracy*4/5;
//--- déplaçons le point du rattachement
   for(int i=0;i<v_steps;i++)
     {
      //--- prenons la valeur suivante
      if(p>1)
         p-=1;
      //--- deplaçons le point
      if(!ArrowRightPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons la marque du graphique
   ArrowRightPriceDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW_BUY

Le signe "Buy".



L'exemple

Le script suivant crée et déplace le signe "Buy" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script dessine le signe \"Buy\" dans la fenêtre du graphique
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input color InpColor=C'3,95,172'; // La couleur des signes
//+------------------------------------------------------------------+
//| Crée le signe "Buy"                                              |
//+------------------------------------------------------------------+
bool ArrowBuyCreate(const long            chart_ID=0,        // ID du graphique
                    const string          name="ArrowBuy",   // le nom du signe
                    const int             sub_window=0,      // le numéro du sous-fené
                    datetime              time=0,            // le temps du point du r
                    double                price=0,           // le prix du point du ra
                    const color           clr=C'3,95,172',   // la couleur du signe
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la ligne
                    const int             width=1,           // la taille de la ligne
                    const bool            back=false,        // à l'arrière-plan
                    const bool            selection=false,   // Sélectionner pour les
```

© 2000-2015, MetaQuotes Software Corp.

```
                    const bool            hidden=true,       // est caché dans la list
                    const long            z_order=0)         // la priorité au clic d'
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécif
   ChangeArrowEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_BUY,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"Buy\"! Le code de l'erreur = ",GetLastError
      return(false);
     }
//--- établissons la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de la ligne (à la sélection)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons la taille de la ligne (à la sélection)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement du signe par la so
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
bool ArrowBuyMove(const long   chart_ID=0,       // ID du graphique
                  const string name="ArrowBuy", // le nom de l'objet
                  datetime     time=0,          // la coordonnée du temps du point du
                  double       price=0)         // la coordonnée du prix du point du r
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le signe "Buy"                                          |
//+------------------------------------------------------------------+
bool ArrowBuyDelete(const long   chart_ID=0,      // ID du graphique
                    const string name="ArrowBuy") // le nom du signe
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le signe\"Buy\"! Le code de l'erreur = ",
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // le tableau pour stocker les dates des barres visibles
   double   low[];  // le tableau pour stocker les prix Low des barres visibles
   double   high[]; // le tableau pour stocker les prix High des barres visibles
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs des prix Low! Le code de l'erreur
      return;
     }
//--- remplissons le tableau des prix High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs des prix High! Le code de l'erreu
      return;
     }
//--- créons les signes "Buy" dans le point Low pour chaque barre visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyCreate(0,"ArrowBuy_"+(string)i,0,date[i],low[i],InpColor))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- déplaçons les signes "Buy" au point High pour chaque barre visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyMove(0,"ArrowBuy_"+(string)i,date[i],high[i]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- supprimons les signes "Buy"
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyDelete(0,"ArrowBuy_"+(string)i))
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW_SELL

Le signe "Sell".

L'exemple

Le script suivant crée et déplace le signe "Sell" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script dessine le signe \"Sell\" dans la fenêtre du graphiqu
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input color InpColor=C'225,68,29'; // La couleur des signes
//+------------------------------------------------------------------+
//| Crée le signe "Sell"                                             |
//+------------------------------------------------------------------+
bool ArrowSellCreate(const long            chart_ID=0,        // ID du graphique
                     const string          name="ArrowSell",  // le nom du signe
                     const int             sub_window=0,      // le numéro du sous-fer
                     datetime              time=0,            // le temps du point du
                     double                price=0,           // le prix du point du
                     const color           clr=C'225,68,29',  // la couleur du signe
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la ligne
                     const int             width=1,           // la taille de la ligne
                     const bool            back=false,        // à l'arrière-plan
                     const bool            selection=false,   // Sélectionner pour les
```

© 2000-2015, MetaQuotes Software Corp.

```
                      const bool            hidden=true,       // est caché dans la lis
                      const long            z_order=0)         // la priorité au clic d
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécif
   ChangeArrowEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le signe
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_SELL,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le signe \"Sell\"! Le code de l'erreur = ",Ge
      return(false);
     }
//--- établissons la couleur du signe
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de la ligne (à la sélection)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons la taille de la ligne (à la sélection)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement du signe par la so
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
bool ArrowSellMove(const long   chart_ID=0,       // ID du graphique
                   const string name="ArrowSell", // le nom de l'objet
                   datetime     time=0,           // la coordonnée du temps du point d
                   double       price=0)          // la coordonnée du prix du point du
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le signe "Sell"                                         |
//+------------------------------------------------------------------+
bool ArrowSellDelete(const long   chart_ID=0,       // ID du graphique
                     const string name="ArrowSell") // le nom du signe
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le signe
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le signe \"Sell\"! Le code de l'erreur =
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // le tableau pour stocker les dates des barres visibles
   double   low[];  // le tableau pour stocker les prix Low des barres visibles
   double   high[]; // le tableau pour stocker les prix High des barres visibles
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs des prix Low! Le code de l'erreur
      return;
     }
//--- remplissons le tableau des prix High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs des prix High! Le code de l'erreu
      return;
     }
//--- créons les signes "Sell" dans le point High pour chaque barre visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellCreate(0,"ArrowSell_"+(string)i,0,date[i],high[i],InpColor))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- déplaçons les signes "Sell" au point Low pour chaque barre visible
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellMove(0,"ArrowSell_"+(string)i,date[i],low[i]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//---supprimons les signes "Sell"
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellDelete(0,"ArrowSell_"+(string)i))
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_ARROW

L'objet "La flèche".

Note

La position du point du rattachement par rapport la flèche peut être choisie de l'énumération ENUM_ARROW_ANCHOR.

On peut créer les flèches d'une grande taille (plus que 5) ayant établi seulement la valeur correspondante de la propriété OBJPROP_WIDTH à l'écriture du code dans MetaEditor.

On peut choisir le type nécessaire de la flèche, ayant établi un des codes des symboles de la fonte Wingdings.

L'exemple

Le script suivant crée l'objet "La flèche" sur le graphique et change son type. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script crée la flèche optionnelle dans la fenêtre du graphic
#property description "La coordonnée du point du rattachement est spécifiée en pourcer
#property description "de la dimension de la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string            InpName="Arrow";        // Le nom de la flèche
input int               InpDate=50;             // La date du point du rattachement er
```

© 2000-2015, MetaQuotes Software Corp.

```
input int                InpPrice=50;             // Le prix du point du rattachement en
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;    // Le moyen du rattachement
input color              InpColor=clrDodgerBlue; // La couleur de la flèche
input ENUM_LINE_STYLE    InpStyle=STYLE_SOLID;    // Le style de la ligne bordant
input int                InpWidth=10;             // La taille de la flèche
input bool               InpBack=false;           // La flèche à l'arrière-plan
input bool               InpSelection=false;      // Sélectionner pour les déplacements
input bool               InpHidden=true;          // Est caché dans la liste des objets
input long               InpZOrder=0;             // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée la flèche                                                   |
//+------------------------------------------------------------------+
bool ArrowCreate(const long              chart_ID=0,           // ID du graphique
                 const string            name="Arrow",         // le nom de la flèche
                 const int               sub_window=0,         // le numéro du sous-fe
                 datetime                time=0,               // le temps du point du
                 double                  price=0,              // le prix du point du
                 const uchar             arrow_code=252,       // le code de la flèche
                 const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // la position du point
                 const color             clr=clrRed,           // la couleur de la flè
                 const ENUM_LINE_STYLE   style=STYLE_SOLID,    // le style de la ligne
                 const int               width=3,              // la taille de la flèc
                 const bool              back=false,           // à l'arrière-plan
                 const bool              selection=true,       // Sélectionner pour le
                 const bool              hidden=true,          // est caché dans la li
                 const long              z_order=0)            // la priorité au clic
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécifi
   ChangeArrowEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons la flèche
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer la flèche! Le code de l'erreur = ",GetLastErr
      return(false);
     }
//--- établissons le code de la flèche
   ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,arrow_code);
//--- établissons le moyen du rattachement
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- établissons la couleur de la flèche
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne bordant
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons la taille de la flèche
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement de la flèche par l
//--- à la création de l'objet graphique par la fonction ObjectCreate, par défaut on n
//--- l'objet. A l'intérieur de cette méthode, le paramètre selection
//--- par défaut est égal à la valeur true, ce qui permet de sélectionner et de déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
bool ArrowMove(const long   chart_ID=0,   // ID du graphique
               const string name="Arrow", // le nom de l'objet
               datetime     time=0,       // la coordonnée du temps du point du rattac
               double       price=0)      // la coordonnée du prix du point du rattach
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Chage le code de la flèche                                       |
//+------------------------------------------------------------------+
bool ArrowCodeChange(const long   chart_ID=0,   // ID du graphique
                     const string name="Arrow", // le nom de l'objet
                     const uchar  code=252)     // le code de la flèche
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons le code de la flèche
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,code))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le code de la flèche! Le code de l'erreur =
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le moyen du rattachement                                  |
//+------------------------------------------------------------------+
bool ArrowAnchorChange(const long              chart_ID=0,        // ID du graphique
                       const string            name="Arrow",      // le nom de l'objet
                       const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // le moyen du ratta
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- chageons le moyen du rattachement
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le moyen du rattachement! Le code de l'erre
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime la flèche                                               |
//+------------------------------------------------------------------+
bool ArrowDelete(const long   chart_ID=0,   // ID du graphique
                 const string name="Arrow") // le nom de la flèche
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la flèche
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer la flèche! Le code de l'erreur = ",GetLas
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- la taille du tableau price
   int accuracy=1000;
//--- les tableaux pour stocker les valeurs des dates et les prix qui seront utilisées
//--- pour l'établissement et le changement des coordonnées du point du rattachement d
   datetime date[];
   double   price[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix
//--- trouvons les valeurs maximale et minimale du graphique
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- définirons le pas du changement du prix et remplirons le tableau
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- définissons les points pour le dessin de la flèche
   int d=InpDate*(bars-1)/100;
```

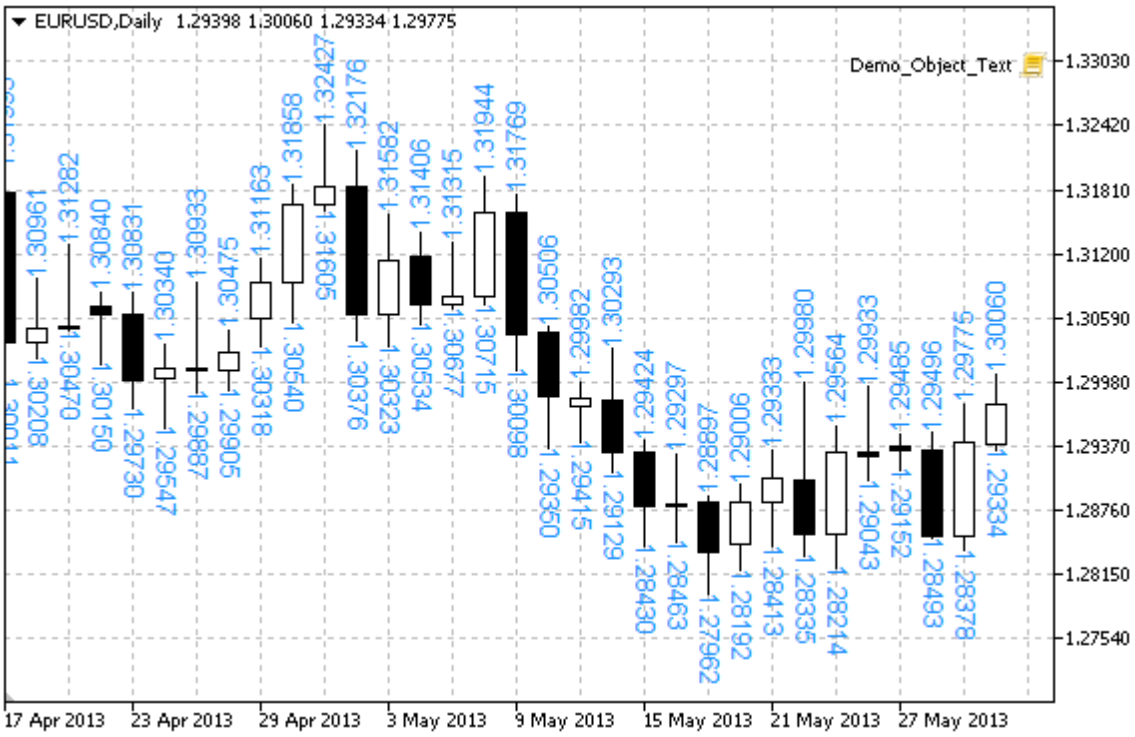
© 2000-2015, MetaQuotes Software Corp.

```
   int p=InpPrice*(accuracy-1)/100;
//--- créons la flèche sur le graphique
   if(!ArrowCreate(0,InpName,0,date[d],price[p],32,InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique
   ChartRedraw();
//--- examinerons toutes les variantes de la construction des flèches dans le cycle
   for(int i=33;i<256;i++)
     {
      if(!ArrowCodeChange(0,InpName,(uchar)i))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à une demi-seconde
      Sleep(500);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons la flèche du graphique
   ArrowDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_TEXT

L'objet "Texte".



Note

La position du point du rattachement par rapport le texte peut être choisie de l'énumération ENUM_ANCHOR_POINT. On peut changer aussi l'angle de l'inclinaison du texte à l'aide de la propriétéOBJPROP_ANGLE.

L'exemple

Le script suivant crée quelques objets "Texte" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script crée l'objet graphique \"Texte\"."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string             InpFont="Arial";         // La fonte
input int                InpFontSize=10;          // La taille de la fonte
input color              InpColor=clrRed;         // La couleur
input double             InpAngle=90.0;           // L'angle de l'inclinaison en degrés
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_BOTTOM; // Le moyen du rattachement
input bool               InpBack=false;           // L'objet à l'arrière-plan
input bool               InpSelection=false;      // Sélectionner pour les déplacements
input bool               InpHidden=true;          // Est caché dans la liste des objets
```

© 2000-2015, MetaQuotes Software Corp.

```
input long              InpZOrder=0;             // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée l'objet "Texte"                                             |
//+------------------------------------------------------------------+
bool TextCreate(const long              chart_ID=0,               // ID du graphique
                const string            name="Text",              // le nom de l'objet
                const int               sub_window=0,             // le numéro du sous
                datetime                time=0,                   // le temps du point
                double                  price=0,                  // le prix du point
                const string            text="Text",              // le texte lui-même
                const string            font="Arial",             // la fonte
                const int               font_size=10,             // la taille de la f
                const color             clr=clrRed,               // la couleur
                const double            angle=0.0,                // l'inclinaison du
                const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // le moyen du ratta
                const bool              back=false,               // à l'arrière-plan
                const bool              selection=false,          // sélectionner pour
                const bool              hidden=true,              // est caché dans la
                const long              z_order=0)                // la priorité au cl
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécif
   ChangeTextEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//---  créons l'objet "Texte"
   if(!ObjectCreate(chart_ID,name,OBJ_TEXT,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer l'objet \"Texte\"! Le code de l'erreur = ",Ge
      return(false);
     }
//--- établissons le texte
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- établissons la fonte du texte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- établissons la taille de la fonte
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- établissons l'angle de l'inclinaison du texte
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- établissons le moyen du rattachement
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- établissons la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement de l'objet par la
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le point du rattachement                                 |
//+------------------------------------------------------------------+
bool TextMove(const long   chart_ID=0,  // ID du graphique
              const string name="Text", // le nom de l'objet
              datetime     time=0,      // la coordonnée du temps du point du rattache
              double       price=0)     // la coordonnée du prix du point du rattacher
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'er
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le texte de l'objet                                       |
//+------------------------------------------------------------------+
bool TextChange(const long   chart_ID=0,  // ID du graphique
                const string name="Text", // le nom de l'objet
                const string text="Text") // le texte
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons le texte de l'objet
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le texte! Le code de l'erreur = ",GetLastEr
      return(false);
     }
//--- l'exécution réussie
   return(true);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Supprime l'objet "Texte"                                         |
//+------------------------------------------------------------------+
bool TextDelete(const long   chart_ID=0,  // ID du graphique
                const string name="Text") // le nom de l'objet
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons l'objet
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer l'objet \"Texte\"! Le code de l'erreur =
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeTextEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // le tableau pour stocker les dates des barres visibles
   double   low[];  // le tableau pour stocker les prix Low des barres visibles
   double   high[]; // le tableau pour stocker les prix High des barres visibles
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
       return;
      }
//--- remplissons le tableau des prix Low
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
      {
       Print("On n'a pas réussi à copier les valeurs des prix Low! Le code de l'erreur
       return;
      }
//--- remplissons le tableau des prix High
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
      {
       Print("On n'a pas réussi à copier les valeurs des prix High! Le code de l'erreu
       return;
      }
//---définissons la fréquence des inscriptions
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- définissons le pas
   int step=1;
   switch(scale)
      {
       case 0:
          step=12;
          break;
       case 1:
          step=6;
          break;
       case 2:
          step=4;
          break;
       case 3:
          step=2;
          break;
      }
//---créons les inscriptions pour les valeurs High et Low des barres (avec les interva
   for(int i=0;i<bars;i+=step)
      {
       //--- créons les inscriptions
       if(!TextCreate(0,"TextHigh_"+(string)i,0,date[i],high[i],DoubleToString(high[i],
          InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
         {
          return;
         }
       if(!TextCreate(0,"TextLow_"+(string)i,0,date[i],low[i],DoubleToString(low[i],5),
          InpColor,-InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
         {
          return;
         }
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à une demi-seconde
   Sleep(500);
//--- supprimons les inscriptions
   for(int i=0;i<bars;i+=step)
     {
      if(!TextDelete(0,"TextHigh_"+(string)i))
         return;
      if(!TextDelete(0,"TextLow_"+(string)i))
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_LABEL

L'objet "La marque de texte".

Note

La position du point du rattachement par rapport la marque peut être choisie de l'énumération ENUM_ANCHOR_POINT. Les coordonnées des points du rattachement sont spécifiées en pixels.

On peut changer aussi l'angle de l'inclinaison de la marque de texte à l'aide de la propriété ENUM_BASE_CORNER.

L'exemple

Le script suivant crée et déplace l'objet "La marque de texte" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script crée l'objet graphique \"La marque de texte.\"."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string            InpName="Label";         // Le nom de la marque
input int               InpX=150;                // La distance selon l'axe X
input int               InpY=150;                // La distance selon l'axe Y
input string            InpFont="Arial";         // La fonte
input int               InpFontSize=14;          // La taille de la fonte
input color             InpColor=clrRed;         // La couleur
input double            InpAngle=0.0;            // L'angle de l'inclinaison en degrés
```

© 2000-2015, MetaQuotes Software Corp.

```
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER; // Le moyen du rattachement
input bool              InpBack=false;           // L'objet à l'arrière-plan
input bool              InpSelection=true;       // Sélectionner pour les déplacements
input bool              InpHidden=true;          // Est caché dans la liste des objets
input long              InpZOrder=0;             // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée la marque de texte                                          |
//+------------------------------------------------------------------+
bool LabelCreate(const long              chart_ID=0,               // ID du graphique
                 const string            name="Label",             // le nom de la mar
                 const int               sub_window=0,             // le numéro du sou
                 const int               x=0,                      // la coordonnée se
                 const int               y=0,                      // la coordonnée se
                 const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // l'angle du graph
                 const string            text="Label",             // le texte
                 const string            font="Arial",             // la fonte
                 const int               font_size=10,             // la taille de la
                 const color             clr=clrRed,               // la couleur
                 const double            angle=0.0,                // l'inclinaison du
                 const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // le moyen du ratt
                 const bool              back=false,               // à l'arrière-plan
                 const bool              selection=false,          // sélectionner pou
                 const bool              hidden=true,              // est caché dans l
                 const long              z_order=0)                // la priorité au c
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons la marque de texte
   if(!ObjectCreate(chart_ID,name,OBJ_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer la marque de texte! Le code de l'erreur = ",G
      return(false);
     }
//---  établissons les coordonnées de la marque
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//---établissons l'angle du graphique, relativement lequel les coordonnées du point se
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- établissons le texte
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- établissons la fonte du texte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- établissons la taille de la fonte
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- établissons l'angle de l'inclinaison du texte
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- établissons le moyen du rattachement
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- établissons la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement de la marque par l
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace la marque de texte                                       |
//+------------------------------------------------------------------+
bool LabelMove(const long   chart_ID=0,   // ID du graphique
               const string name="Label", // le nom de la marque
               const int    x=0,          // la coordonnée selon l'axe X
               const int    y=0)          // la coordonnée selon l'axe Y
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons la marque de texte
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée X de la marque! Le code de l
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée Y de la marque! Le code de l
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change l'angle du graphique pour le rattachement de la marque
//+------------------------------------------------------------------+
bool LabelChangeCorner(const long             chart_ID=0,               // ID du graph
                       const string           name="Label",             // le nom de l
                       const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // l'angle du
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- changeons l'angle du rattachement
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer l'angle du rattachement! Le code de l'erreu
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le texte de l'objet                                       |
//+------------------------------------------------------------------+
bool LabelTextChange(const long   chart_ID=0,   // ID du graphique
                     const string name="Label", // le nom de l'objet
                     const string text="Text")  // le texte
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons le texte de l'objet
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le texte! Le code de l'erreur = ",GetLastEr
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime la marque de texte                                      |
//+------------------------------------------------------------------+
bool LabelDelete(const long   chart_ID=0,   // ID du graphique
                 const string name="Label") // le nom de la marque
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la marque
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer la marque de texte! Le code de l'erreur =
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- retiendrons les coordonnées de la marque aux variables locales
   int x=InpX;
   int y=InpY;
//--- les tailles de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définirons les tailles de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("On n'a pas réussi à recevoir la largeur du graphique! Le code de l'erreu
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("On n'a pas réussi à recevoir la hauteur du graphique! Le code de l'erreu
      return;
     }
//--- vérifions les paramètres d'entrée à la validité
   if(InpX<0 || InpX>x_distance-1 || InpY<0 || InpY>y_distance-1)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- préparerons le texte initial pour la marque
   string text;
   StringConcatenate(text,"Un angle gauche supérieur: ",x,",",y);
//--- créons la marque de texte sur le graphique
   if(!LabelCreate(0,InpName,0,InpX,InpY,CORNER_LEFT_UPPER,text,InpFont,InpFontSize,
      InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- copions le graphique et attendrons une demi-seconde
   ChartRedraw();
   Sleep(500);
//--- déplaçons la marque et changeons simultanément son texte
//--- le nombre d'itérations selon les axes
   int h_steps=(int)(x_distance/2-InpX);
   int v_steps=(int)(y_distance/2-InpY);
//--- déplaçons la marque en bas
   for(int i=0;i<v_steps;i++)
     {
      //--- changeons la coordonnée
      y+=2;
      //--- déplaçons la marque et changeons son texte
      MoveAndTextChange(x,y,"Un angle gauche supérieur: ");
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- le retard à une demi-seconde
   Sleep(500);
//--- déplaçons la marque à droit
   for(int i=0;i<h_steps;i++)
     {
      //--- changeons la coordonnée
      x+=2;
      //--- déplaçons la marque et changeons son texte
      MoveAndTextChange(x,y,"Un angle gauche supérieur: ");
     }
//--- le retard à une demi-seconde
   Sleep(500);
//--- déplaçons la marque en haut
   for(int i=0;i<v_steps;i++)
     {
      //--- changeons la coordonnée
      y-=2;
      //--- déplaçons la marque et changeons son texte
      MoveAndTextChange(x,y,"Un angle gauche supérieur: ");
     }
//--- le retard à une demi-seconde
   Sleep(500);
//--- déplaçons la marque à gauche
   for(int i=0;i<h_steps;i++)
     {
      //--- changeons la coordonnée
      x-=2;
      //--- déplaçons la marque et changeons son texte
      MoveAndTextChange(x,y,"Un angle gauche supérieur: ");
     }
//--- le retard à une demi-seconde
   Sleep(500);
//--- maintenant déplaçons le point en changeant l'angle du rattachement
//--- déplaçons à l'angle gauche inférieur
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_LOWER))
      return;
//--- changeons le texte de la marque
   StringConcatenate(text,"L'angle gauche inférieur: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- copions le graphique et attendrons deux secondes
   ChartRedraw();
   Sleep(2000);
//---déplaçons à l'angle droit inférieur
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_LOWER))
      return;
//--- changeons le texte de la marque
   StringConcatenate(text,"L'angle droit inférieur: ",x,",",y);
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!LabelTextChange(0,InpName,text))
      return;
//--- copions le graphique et attendrons deux secondes
   ChartRedraw();
   Sleep(2000);
//--- déplaçons à l'angle droit supérieur
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_UPPER))
      return;
//--- changeons le texte de la marque
   StringConcatenate(text,"L'angle droit supérieur: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- copions le graphique et attendrons deux secondes
   ChartRedraw();
   Sleep(2000);
//--- déplaçons à l'angle gauche supérieur
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_UPPER))
      return;
//--- changeons le texte de la marque
   StringConcatenate(text,"Un angle gauche supérieur: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- copions le graphique et attendrons deux secondes
   ChartRedraw();
   Sleep(2000);
//--- supprimons la marque
   LabelDelete(0,InpName);
//--- copions le graphique et attendrons une demi-seconde
   ChartRedraw();
   Sleep(500);
//---
  }
//+------------------------------------------------------------------+
//| La fonction déplace l'objet et change son texte                  |
//+------------------------------------------------------------------+
bool MoveAndTextChange(const int x,const int y,string text)
  {
//--- déplaçons la marque
   if(!LabelMove(0,InpName,x,y))
      return(false);
//--- changeons le texte de la marque
   StringConcatenate(text,text,x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return(false);
//--- vérifions le fait de l'arrêt forcé du script
   if(IsStopped())
      return(false);
//--- redessinons le graphique
   ChartRedraw();
```

© 2000-2015, MetaQuotes Software Corp.

```
// le retard à 0.01 secondes
   Sleep(10);
//--- la sortie de la fonction
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_BUTTON

L'objet "Le bouton".

Note

Les coordonnées des points du rattachement sont spécifiées en pixels. On peut choisir l'angle du rattachement du bouton de l'enumération. ENUM_BASE_CORNER.

L'exemple

Le script suivant crée et déplace l'objet "Le bouton" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script crée le bouton sur le graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string           InpName="Button";           // Le nom du bouton
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // L'angle du graphique pour le ra
input string           InpFont="Arial";            // La fonte
input int              InpFontSize=14;             // La taille de la fonte
input color            InpColor=clrBlack;          // La couleur du texte
input color            InpBackColor=C'236,233,216'; // La couleur du fond
input color            InpBorderColor=clrNONE;     // La couleur de la frontière
input bool             InpState=false;             // Est appuyée/est pressé
input bool             InpBack=false;              // L'objet à l'arrière-plan
input bool             InpSelection=false;         // Sélectionner pour les déplaceme
```

© 2000-2015, MetaQuotes Software Corp.

```
input bool              InpHidden=true;              // Est caché dans la liste des obj
input long              InpZOrder=0;                 // La priorité au clic d'une souri
//+------------------------------------------------------------------+
//| Crée le bouton                                                   |
//+------------------------------------------------------------------+
bool ButtonCreate(const long              chart_ID=0,               // ID du graphique
                  const string            name="Button",            // le nom du bouto
                  const int               sub_window=0,             // le numéro du so
                  const int               x=0,                      // la coordonnée s
                  const int               y=0,                      // la coordonnée s
                  const int               width=50,                 // la largeur du b
                  const int               height=18,                // la hauteur du b
                  const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // l'angle du grap
                  const string            text="Button",            // le texte
                  const string            font="Arial",             // la fonte
                  const int               font_size=10,             // la taille de la
                  const color             clr=clrBlack,             // la couleur du t
                  const color             back_clr=C'236,233,216',  // la couleur du f
                  const color             border_clr=clrNONE,       // la couleur de l
                  const bool              state=false,              // est appuyée/est
                  const bool              back=false,               // à l'arrière-pla
                  const bool              selection=false,          // Sélectionner po
                  const bool              hidden=true,              // est caché dans
                  const long              z_order=0)                // la priorité au
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le boutons
   if(!ObjectCreate(chart_ID,name,OBJ_BUTTON,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le bouton! Le code de l'erreur = ",GetLastErr
      return(false);
     }
//---  établissons les coordonnées du bouton
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//---  établissons la taille du bouton
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//---établissons l'angle du graphique, relativement lequel les coordonnées du point se
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- établissons le texte
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- établissons la fonte du texte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- établissons la taille de la fonte
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- établissons la couleur du texte
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons la couleur du fond
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- établissons la couleur de la frontière
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- set button state
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- activons (true) ou désactivons (false) le mode du déplacement du bouton par la s
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le bouton                                                |
//+------------------------------------------------------------------+
bool ButtonMove(const long   chart_ID=0,    // ID du graphique
                const string name="Button", // le nom du bouton
                const int    x=0,           // la coordonnée selon l'axe X
                const int    y=0)           // la coordonnée selon l'axe Y
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le bouton
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée X du bouton! Le code de l'er
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée Y du bouton! Le code de l'er
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change la taille du bouton                                       |
//+------------------------------------------------------------------+
bool ButtonChangeSize(const long   chart_ID=0,    // ID du graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
                      const string name="Button", // le nom du bouton
                      const int    width=50,      // la largeur du bouton
                      const int    height=18)     // la hauteur du bouton
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons la taille du bouton
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la largeur du bouton! Le code de l'erreur =
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le hauteur du bouton! Le code de l'erreur =
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change l'angle du graphique pour le rattachement du bouton       |
//+------------------------------------------------------------------+
bool ButtonChangeCorner(const long             chart_ID=0,          // ID du grap
                        const string           name="Button",       // le nom du
                        const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // l'angle du
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons l'angle du rattachement
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer l'angle du rattachement! Le code de l'erreu
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le texte du bouton                                        |
//+------------------------------------------------------------------+
bool ButtonTextChange(const long   chart_ID=0,     // ID du graphique
                      const string name="Button", // le nom du bouton
                      const string text="Text")   // le texte
  {
//--- oblitérons la valeur de l'erreur
```

© 2000-2015, MetaQuotes Software Corp.

```
   ResetLastError();
//--- changeons le texte de l'objet
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le texte! Le code de l'erreur = ",GetLastEr
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le bouton                                               |
//+------------------------------------------------------------------+
bool ButtonDelete(const long   chart_ID=0,    // ID du graphique
                  const string name="Button") // le nom du bouton
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le bouton
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le bouton! Le code de l'erreur = ",GetLas
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- les tailles de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définirons les tailles de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("On n'a pas réussi à recevoir la largeur du graphique! Le code de l'erreur
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("On n'a pas réussi à recevoir la hauteur du graphique! Le code de l'erreur
      return;
     }
//--- définirons le pas pour le changement de la taille du bouton
```

© 2000-2015, MetaQuotes Software Corp.

```
   int x_step=(int)x_distance/32;
   int y_step=(int)y_distance/32;
//---établissons les coordonnées du bouton et sa taille
   int x=(int)x_distance/32;
   int y=(int)y_distance/32;
   int x_size=(int)x_distance*15/16;
   int y_size=(int)y_distance*15/16;
//--- créons le boutons
   if(!ButtonCreate(0,InpName,0,x,y,x_size,y_size,InpCorner,"Press",InpFont,InpFontSiz
      InpColor,InpBackColor,InpBorderColor,InpState,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
//--- redessinons le graphique
   ChartRedraw();
//--- diminuerons le bouton dans le cycle
   int i=0;
   while(i<13)
     {
      //--- le retard à une demi-seconde
      Sleep(500);
      //--- passons le bouton à l'état appuyé
      ObjectSetInteger(0,InpName,OBJPROP_STATE,true);
      //--- copions le graphique et attendrons 0.2 secondes
      ChartRedraw();
      Sleep(200);
      //--- redéfinirons les coordonnées et la taille du bouton
      x+=x_step;
      y+=y_step;
      x_size-=x_step*2;
      y_size-=y_step*2;
      //---diminuerons le bouton
      ButtonMove(0,InpName,x,y);
      ButtonChangeSize(0,InpName,x_size,y_size);
      //--- rendons le bouton à l'état non appuyé
      ObjectSetInteger(0,InpName,OBJPROP_STATE,false);
      //---redessinons le graphique
      ChartRedraw();
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //--- augmentons le compteur du cycle
      i++;
     }
//--- le retard à une demi-seconde
   Sleep(500);
//--- supprimons le bouton
   ButtonDelete(0,InpName);
   ChartRedraw();
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- attendons 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_CHART

L'objet "Graphique".

Note

Les coordonnées des points du rattachement sont spécifiées en pixels. On peut choisir l'angle du rattachement de l'enumération ENUM_BASE_CORNER.

On peut choisir le symbole, la période et l'échelle, ainsi d'activer/désactiver le mode de l'affichage de l'échelle du prix et de la date pour l'objet "le Graphique".

L'exemple

Le script suivant crée et déplace l'objet "Graphique" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script crée l'objet \"Graphique\"."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string           InpName="Chart";              // Le nom de l'objet
input string           InpSymbol="EURUSD";           // Le symbole
input ENUM_TIMEFRAMES  InpPeriod=PERIOD_H1;          // La période
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // L'angle pour le rattachement
input int              InpScale=2;                   // L'échelle
input bool             InpDateScale=true;            // L'affichage de l'échelle du ter
input bool             InpPriceScale=true;           // L'affichage de l'échelle du pri
```

© 2000-2015, MetaQuotes Software Corp.

```
input color            InpColor=clrRed;             // La couleur du cadre à la sélect
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT;   // Le style de la ligne à la séle
input int              InpPointWidth=1;             // La taille du point à déplacer
input bool             InpBack=false;               // L'objet à l'arrière-plan
input bool             InpSelection=true;           // Sélectionner pour les déplaceme
input bool             InpHidden=true;              // Est caché dans la liste des obj
input long             InpZOrder=0;                 // La priorité au clic d'une souri
//+------------------------------------------------------------------+
//| Crée l'objet "le Graphique"                                      |
//+------------------------------------------------------------------+
bool ObjectChartCreate(const long              chart_ID=0,               // ID du grap
                       const string            name="Chart",             // le nom de
                       const int               sub_window=0,             // le numéro
                       const string            symbol="EURUSD",          // le symbole
                       const ENUM_TIMEFRAMES   period=PERIOD_H1,         // la période
                       const int               x=0,                      // la coordon
                       const int               y=0,                      // la coordon
                       const int               width=300,                // la largeur
                       const int               height=200,               // la hauteur
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // l'angle du
                       const int               scale=2,                  // l'échelle
                       const bool              date_scale=true,          // l'affichag
                       const bool              price_scale=true,         // l'affichag
                       const color             clr=clrRed,               // la couleur
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // le style d
                       const int               point_width=1,            // la taille
                       const bool              back=false,               // à l'arrièr
                       const bool              selection=false,          // Sélectionn
                       const bool              hidden=true,              // est caché
                       const long              z_order=0)                // la priorit
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons l'objet "Graphique"
   if(!ObjectCreate(chart_ID,name,OBJ_CHART,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer l'objet \"Graphique\"! Le code de l'erreur =
      return(false);
     }
//---établissons les coordonnées de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- établissons la taille de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//---établissons l'angle du graphique, relativement lequel les coordonnées du point se
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- établissons le symbole
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol);
//---  établissons la période
   ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period);
//--- établissons l'échelle
   ObjectSetInteger(chart_ID,name,OBJPROP_CHART_SCALE,scale);
//--- affichons (true) ou cachons (false) l'échelle du temps
   ObjectSetInteger(chart_ID,name,OBJPROP_DATE_SCALE,date_scale);
//---affichons (true) ou cachons (false) l'échelle du prix
   ObjectSetInteger(chart_ID,name,OBJPROP_PRICE_SCALE,price_scale);
//--- établissons la couleur du cadre au mode inséré de la sélection de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne du cadre au mode inséré de la sélection de l'obj
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//---établissons la taille du point du rattachement, qui peut être utilisé pour déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement de la marque par l
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Établit le symbole et le temps trame de l'objet "le Graphique"   |
//+------------------------------------------------------------------+
bool ObjectChartSetSymbolAndPeriod(const long            chart_ID=0,       // ID du gr
                                   const string          name="Chart",     // le nom d
                                   const string          symbol="EURUSD",  // le symbo
                                   const ENUM_TIMEFRAMES period=PERIOD_H1) // le temps
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons le symbole et le temps trame de l'objet "le Graphique"
   if(!ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à établir le symbole pour l'objet \"Graphique\"! Le c
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à établir la période pour l'objet \"Graphique\"! Le c
      return(false);
```

© 2000-2015, MetaQuotes Software Corp.

```
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace l'objet "le Graphique"                                   |
//+------------------------------------------------------------------+
bool ObjectChartMove(const long   chart_ID=0,   // ID du graphique (non de l'objet)
                     const string name="Chart", // le nom de l'objet
                     const int    x=0,          // la coordonnée selon l'axe X
                     const int    y=0)          // la coordonnée selon l'axe Y
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée X de l'objet \"le Graphique\
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée Y de l'objet \"le Graphique\
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change la taille de l'objet "le Graphique"                       |
//+------------------------------------------------------------------+
bool ObjectChartChangeSize(const long   chart_ID=0,   // ID du graphique (non de l'obj
                           const string name="Chart", // le nom de l'objet
                           const int    width=300,    // la largeur
                           const int    height=200)   // la hauteur
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons la taille de l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la largeur de l'objet \"le Graphique\"! Le
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le hauteur de l'objet \"le Graphique\"! Le
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Rend ID de l'objet "le Graphique"                                |
//+------------------------------------------------------------------+
long ObjectChartGetID(const long   chart_ID=0,   // ID du graphique (non de l'objet)
                      const string name="Chart") // le nom de l'objet
  {
//--- préparons la variable pour la réception de l'ID de l'objet "le Graphique"
   long id=-1;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons ID
   if(!ObjectGetInteger(chart_ID,name,OBJPROP_CHART_ID,0,id))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à recevoir ID de l'objet \"le Graphique\"! Le code de
     }
//--- le retour du résultat
   return(id);
  }
//+------------------------------------------------------------------+
//| Supprime l'objet "le Graphique"                                  |
//+------------------------------------------------------------------+
bool ObjectChartDelete(const long   chart_ID=0,   // ID du graphique (non de l'objet)
                       const string name="Chart") // le nom de l'objet
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons le bouton
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer l'objet \"Graphique\"! Le code de l'erreu
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//---recevrons le nombre de symboles dans "Aperçu du marché"
   int  symbols=SymbolsTotal(true);
//--- définirons, s'il y a le symbole avec le nom indiqué dans la liste des symboles
   bool exist=false;
   for(int i=0;i<symbols;i++)
      if(InpSymbol==SymbolName(i,true))
        {
         exist=true;
         break;
        }
   if(!exist)
     {
      Print("L'erreur! Ce symbole ",InpSymbol," n'est pas présenté dans la fenêtre \"A
      return;
     }
//--- La vérification des paramètres d'entrée de la convenance
   if(InpScale<0 || InpScale>5)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }

//--- les tailles de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définirons les tailles de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("On n'a pas réussi à recevoir la largeur du graphique! Le code de l'erreur
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("On n'a pas réussi à recevoir la hauteur du graphique! Le code de l'erreur
      return;
     }
//--- établissons les coordonnées de l'objet "le Graphique" et son taille
   int x=(int)x_distance/16;
   int y=(int)y_distance/16;
   int x_size=(int)x_distance*7/16;
   int y_size=(int)y_distance*7/16;
//--- créons l'objet "Graphique"
   if(!ObjectChartCreate(0,InpName,0,InpSymbol,InpPeriod,x,y,x_size,y_size,InpCorner,I
      InpPriceScale,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
```

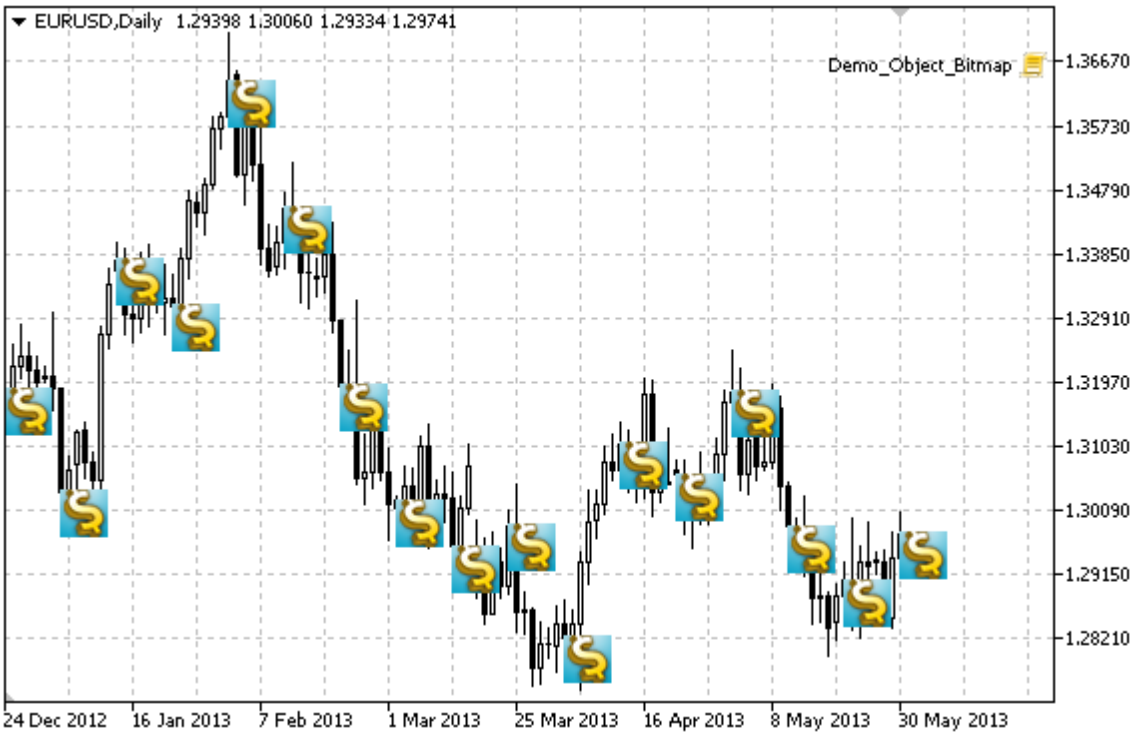
© 2000-2015, MetaQuotes Software Corp.

```
   Sleep(1000);
//--- agrandissons l'objet "le Graphique"
   int steps=(int)MathMin(x_distance*7/16,y_distance*7/16);
   for(int i=0;i<steps;i++)
     {
      //--- changeons la taille
      x_size+=1;
      y_size+=1;
      if(!ObjectChartChangeSize(0,InpName,x_size,y_size))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //--- redessinons le graphique et attendons 0.01 secondes
      ChartRedraw();
      Sleep(10);
     }
//--- le retard à une demi-seconde
   Sleep(500);
//--- changeons le temps trame du graphique
   if(!ObjectChartSetSymbolAndPeriod(0,InpName,InpSymbol,PERIOD_M1))
      return;
   ChartRedraw();
//--- le retard à trois secondes
   Sleep(3000);
//--- supprimons l'objet
   ObjectChartDelete(0,InpName);
   ChartRedraw();
//--- attendons 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_BITMAP

L'objet "le Dessin".



Note

On peut choisirle domaine de la visibilité du dessin pour l'objet "le Dessin".

L'exemple

Le script suivant crée quelques images sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script crée le dessin dans la fenêtre du graphique."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string          InpFile="\\Images\\dollar.bmp"; // Le nom du fichier avec l'imag
input int             InpWidth=24;                    // la coordonnée X du domaine de
input int             InpHeight=24;                   // la coordonnée Y du domaine de
input int             InpXOffset=4;                   // Le déplacement du domaine de
input int             InpYOffset=4;                   // Le déplacement du domaine de
input color           InpColor=clrRed;                // La couleur du cadre à la séle
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID;           // Le style de la ligne à la sél
input int             InpPointWidth=1;                // La taille du point à déplacer
input bool            InpBack=false;                  // L'objet à l'arrière-plan
input bool            InpSelection=false;             // Sélectionner pour les déplace
```

© 2000-2015, MetaQuotes Software Corp.

```
input bool            InpHidden=true;                  // Est caché dans la liste des o
input long            InpZOrder=0;                     // La priorité au clic d'une sou
//+------------------------------------------------------------------+
//| Crée le dessin dans la fenêtre du graphique                      |
//+------------------------------------------------------------------+
bool BitmapCreate(const long            chart_ID=0,        // ID du graphique
                  const string          name="Bitmap",     // le nom du dessin
                  const int             sub_window=0,      // le numéro du sous-fenêtr
                  datetime              time=0,            // le temps du point du rat
                  double                price=0,           // le prix du point du ratt
                  const string          file="",           // le nom du fichier avec l
                  const int             width=10,          // la coordonnée X du domai
                  const int             height=10,         // la coordonnée Y du domai
                  const int             x_offset=0,        // le déplacement du domain
                  const int             y_offset=0,        // le déplacement du domain
                  const color           clr=clrRed,        // la couleur du cadre à la
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // le style de la ligne à l
                  const int             point_width=1,     // la taille du point du dé
                  const bool            back=false,        // à l'arrière-plan
                  const bool            selection=false,   // Sélectionner pour les dé
                  const bool            hidden=true,       // est caché dans la liste
                  const long            z_order=0)         // la priorité au clic d'un
  {
//--- établissons les coordonnées des points du rattachement, s'ils ne sont pas spécif
   ChangeBitmapEmptyPoint(time,price);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le dessin
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer le dessin dans la fenêtre du graphique! Le co
      return(false);
     }
//--- établissons la voie vers le fichier avec l'image
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à charger l'image! Le code de l'erreur = ",GetLastErr
      return(false);
     }
//--- établissons le domaine de la visibilité de l'image; si les valeurs de la largeur
//--- sont plus grandes que les valeurs de la largeur et la hauteur (en conséquence) d
//--- elle n'est pas dessinée; si les valeurs de la largeur et de la hauteur sont plus
//--- on dessine sa partie, qui correspond à cette taille
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//---établissons quelle partie de l'image doit se montrer dans le domaine de la visibi
//---c'est par défaut un domaine supérieur gauche de l'image; les valeurs permettent
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- de faire le décalage de cet angle et afficher une autre partie de l'image
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- établissons la couleur du cadre au mode inséré de la sélection de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne du cadre au mode inséré de la sélection de l'obj
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//---établissons la taille du point du rattachement, qui peut être utilisé pour déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement de la marque par l
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sou
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Établit une nouvelle image pour le dessin                        |
//+------------------------------------------------------------------+
bool BitmapSetImage(const long   chart_ID=0,    // ID du graphique
                    const string name="Bitmap", // le nom du dessin
                    const string file="")       // la voie vers le fichier
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la voie vers le fichier avec l'image
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à charger l'image! Le code de l'erreur = ",GetLastErr
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace le dessin dans la fenêtre du graphique                   |
//+------------------------------------------------------------------+
bool BitmapMove(const long   chart_ID=0,    // ID du graphique
                const string name="Bitmap", // le nom du dessin
                datetime     time=0,        // le temps du point du rattachement
                double       price=0)       // le prix du point du rattachement
  {
//--- si les coordonnées du point ne sont pas spécifiées, déplaçons-la sur la barre co
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons le point du rattachement
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer le point du rattachement! Le code de l'err
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change la taille du domaine de la visibilité (la taille du dessin)
//+------------------------------------------------------------------+
bool BitmapChangeSize(const long   chart_ID=0,     // ID du graphique
                      const string name="Bitmap", // le nom du dessin
                      const int    width=0,       // la largeur du dessin
                      const int    height=0)      // la hauteur du dessin
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons la taille du dessin
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la largeur du dessin! Le code de l'erreur =
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le hauteur du dessin! Le code de l'erreur =
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change la coordonnée de l'angle gauche supérieur du domaine de la visibilité
//+------------------------------------------------------------------+
bool BitmapMoveVisibleArea(const long   chart_ID=0,     // ID du graphique
                           const string name="Bitmap", // le nom du dessin
                           const int    x_offset=0,    // la coordonnée X du domaine d
                           const int    y_offset=0)    // la coordonnée Y du domaine d
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons les coordonnées du domaine de la visibilité du dessin
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la coordonnée X du domaine de la visibilité
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la coordonnée Y du domaine de la visibilité
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime le dessin                                               |
//+------------------------------------------------------------------+
bool BitmapDelete(const long   chart_ID=0,    // ID du graphique
                  const string name="Bitmap") // le nom du dessin
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la marque
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer le dessin! Le code de l'erreur = ",GetLas
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Vérifie les valeurs du point du rattachement du signe, et pour les valeurs
//| vides établit les valeurs par défaut                             |
//+------------------------------------------------------------------+
void ChangeBitmapEmptyPoint(datetime &time,double &price)
  {
//--- si le temps du point n'est pas spécifié, il sera sur la barre courante
   if(!time)
      time=TimeCurrent();
//--- si le prix du point n'est pas spécifié, il aura la valeur Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[];  // le tableau pour stocker les dates des barres visibles
   double   close[]; // le tableau pour stocker les prix Close
//--- le nom du fichier avec l'image
   string   file="\\Images\\dollar.bmp";
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
   ArrayResize(close,bars);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- remplissons le tableau des prix Close
   if(CopyClose(Symbol(),Period(),0,bars,close)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs des prix Close! Le code de l'erreu
      return;
     }
//--- définissons la fréquence de la sortie des images
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- définissons le pas
   int step=1;
   switch(scale)
     {
      case 0:
         step=27;
         break;
      case 1:
         step=14;
         break;
      case 2:
         step=7;
         break;
      case 3:
         step=4;
         break;
      case 4:
         step=2;
         break;
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- créons les dessins pour les valeurs High et Low des barres (avec les intervalles
   for(int i=0;i<bars;i+=step)
     {
      //--- créons les dessins
      if(!BitmapCreate(0,"Bitmap_"+(string)i,0,date[i],close[i],InpFile,InpWidth,InpHe
         InpYOffset,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
        {
         return;
        }
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à une demi-seconde
   Sleep(500);
//---supprimons les signes "Sell"
   for(int i=0;i<bars;i+=step)
     {
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_BITMAP_LABEL

L'objet "La marque graphique".

Note

La position du point du rattachement par rapport la marque peut être choisie de l'énumération ENUM_ANCHOR_POINT. Les coordonnées des points du rattachement sont spécifiées en pixels.

On peut choisir aussi l'angledu rattachement de la marque graphique de l'enumération ENUM_BASE_CORNER.

On peut choisirle domaine de la visibilité du dessin pour l'objet "La marque graphique".

L'exemple

Le script suivant crée quelques images sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script crée l'objet \"La marque graphique\"."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string            InpName="BmpLabel";              // Le nom de la marque
input string            InpFileOn="\\Images\\dollar.bmp"; // Le nom du fichier pour le
input string            InpFileOff="\\Images\\euro.bmp";  // Le nom du fichier pour le
input bool              InpState=false;                   // La marque est appuyée/est
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER;      // L'angle du graphique pour
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER;          // Le moyen du rattachement
```

© 2000-2015, MetaQuotes Software Corp.

```
input color             InpColor=clrRed;                  // La couleur du cadre à la
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID;             // Le style de la ligne à la
input int               InpPointWidth=1;                  // La taille du point à dépl
input bool              InpBack=false;                    // L'objet à l'arrière-plan
input bool              InpSelection=false;               // Sélectionner pour les dép
input bool              InpHidden=true;                   // Est caché dans la liste d
input long              InpZOrder=0;                      // La priorité au clic d'une
//+------------------------------------------------------------------+
//| Crée l'objet "La marque graphique"                               |
//+------------------------------------------------------------------+
bool BitmapLabelCreate(const long              chart_ID=0,               // ID du grap
                       const string            name="BmpLabel",          // le nom de
                       const int               sub_window=0,             // le numéro
                       const int               x=0,                      // la coordon
                       const int               y=0,                      // la coordon
                       const string            file_on="",               // l'image en
                       const string            file_off="",              // l'image en
                       const int               width=0,                  // la coordon
                       const int               height=0,                 // la coordon
                       const int               x_offset=10,              // le déplace
                       const int               y_offset=10,              // Le déplace
                       const bool              state=false,              // est appuyé
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // l'angle du
                       const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // le moyen d
                       const color             clr=clrRed,               // la couleur
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // le style d
                       const int               point_width=1,            // la taille
                       const bool              back=false,               // à l'arrièr
                       const bool              selection=false,          // Sélectionn
                       const bool              hidden=true,              // est caché
                       const long              z_order=0)                // la priorit
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons la marque graphique
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer l'objet \"La marque graphique\"! Le code de l
      return(false);
     }
//--- établissons les images pour les modes On et Off
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,0,file_on))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à charger l'image pour le mode On! Le code de l'erreu
      return(false);
     }
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,1,file_off))
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       Print(__FUNCTION__,
             ": on n'a pas réussi à charger l'image pour le mode Off! Le code de l'erre
       return(false);
      }
//---  établissons les coordonnées de la marque
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- établissons le domaine de la visibilité de l'image; si les valeurs de la largeur
//--- sont plus grandes que les valeurs de la largeur et la hauteur (en conséquence) d
//--- elle n'est pas dessinée; si les valeurs de la largeur et de la hauteur sont plus
//--- on dessine sa partie, qui correspond à cette taille
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//---établissons quelle partie de l'image doit se montrer dans le domaine de la visibi
//---c'est par défaut un domaine supérieur gauche de l'image; les valeurs permettent
//--- de faire le décalage de cet angle et afficher une autre partie de l'image
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- établissons dans quel état se trouve la marque (est appuyée ou est pressée)
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//---établissons l'angle du graphique, relativement lequel les coordonnées du point se
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- établissons le moyen du rattachement
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- établissons la couleur du cadre au mode inséré de la sélection de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//---établissons le style de la ligne du cadre au mode inséré de la sélection de l'obj
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//---établissons la taille du point du rattachement, qui peut être utilisé pour déplac
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement de la marque par l
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Établit une nouvelle image pour l'objet "La marque graphique"    |
//+------------------------------------------------------------------+
bool BitmapLabelSetImage(const long   chart_ID=0,      // ID du graphique
                         const string name="BmpLabel", // le nom de la marque
                         const int    on_off=0,        // le modificateur (On ou Off)
```

© 2000-2015, MetaQuotes Software Corp.

```
                          const string file="")         // la voie vers le fichier
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- établissons la voie vers le fichier avec l'image
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,on_off,file))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à charger l'image! Le code de l'erreur = ",GetLastErr
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace l'objet "La marque graphique"                            |
//+------------------------------------------------------------------+
bool BitmapLabelMove(const long   chart_ID=0,      // ID du graphique
                     const string name="BmpLabel", // le nom de la marque
                     const int    x=0,             // la coordonnée selon l'axe X
                     const int    y=0)             // la coordonnée selon l'axe Y
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée X de l'objet! Le code de l'e
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée Y de l'objet! Le code de l'e
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change la taille du domaine de la visibilité (la taille de l'objet)|
//+------------------------------------------------------------------+
bool BitmapLabelChangeSize(const long   chart_ID=0,      // ID du graphique
                           const string name="BmpLabel", // le nom de la marque
                           const int    width=0,         // la largeur de la marque
                           const int    height=0)        // la hauteur de la marque
  {
//--- oblitérons la valeur de l'erreur
```

© 2000-2015, MetaQuotes Software Corp.

```
   ResetLastError();
//--- changeons la taille de l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la largeur de l'objet! Le code de l'erreur
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le hauteur de l'objet! Le code de l'erreur
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change la coordonnée de l'angle gauche supérieur du domaine de la visibilité
//+------------------------------------------------------------------+
bool BitmapLabelMoveVisibleArea(const long   chart_ID=0,      // ID du graphique
                                const string name="BmpLabel", // le nom de la marque
                                const int    x_offset=0,      // la coordonnée X du do
                                const int    y_offset=0)      // la coordonnée Y du do
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons les coordonnées du domaine de la visibilité de l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la coordonnée X du domaine de la visibilité
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la coordonnée Y du domaine de la visibilité
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime l'objet "La marque graphique"                           |
//+------------------------------------------------------------------+
bool BitmapLabelDelete(const long   chart_ID=0,      // ID du graphique
                       const string name="BmpLabel") // le nom de la marque
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la marque
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer l'objet \"La marque graphique\"! Le code
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- les tailles de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définirons les tailles de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("On n'a pas réussi à recevoir la largeur du graphique! Le code de l'erreu
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("On n'a pas réussi à recevoir la hauteur du graphique! Le code de l'erreu
      return;
     }
//--- définirons les coordonnées de la marque graphique
   int x=(int)x_distance/2;
   int y=(int)y_distance/2;
//---établissons les tailles de la marque et les coordonnées du domaine de la visibili
   int width=32;
   int height=32;
   int x_offset=0;
   int y_offset=0;
//--- plaçons la marque graphique au centre de la fenêtre
   if(!BitmapLabelCreate(0,InpName,0,x,y,InpFileOn,InpFileOff,width,height,x_offset,y_
      InpCorner,InpAnchor,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidd
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- changeons les tailles du domaine de la visibilité de la marque dans le cycle
```

© 2000-2015, MetaQuotes Software Corp.

```
   for(int i=0;i<6;i++)
     {
      //--- changeons les tailles du domaine de la visibilité
      width--;
      height--;
      if(!BitmapLabelChangeSize(0,InpName,width,height))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.03 seconde
      Sleep(300);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- changeons les coordonnées du domaine de la visibilité de la marque dans le cycle
   for(int i=0;i<2;i++)
     {
      //--- changeons les coordonnées du domaine de la visibilité
      x_offset++;
      y_offset++;
      if(!BitmapLabelMoveVisibleArea(0,InpName,x_offset,y_offset))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.03 seconde
      Sleep(300);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons la marque
   BitmapLabelDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_EDIT

L'objet "Le champ de saisie"

Note

Les coordonnées des points du rattachement sont spécifiées en pixels. On peut choisir l'angle du rattachement du "Champ de saisie" de l'énumérationENUM_BASE_CORNER.

On peut aussi choisir un des types de l'alignement du texte au-dedans du "Champ de saisie" de l'énumérationENUM_ALIGN_MODE.

L'exemple

Le script suivant crée et déplace l'objet "Le champ de saisie" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script crée l'objet \""Le champ de saisie\"."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string            InpName="Edit";              // Le nom de l'objet
input string            InpText="Text";              // Le texte de l'objet
input string            InpFont="Arial";             // La fonte
input int               InpFontSize=14;              // La taille de la fonte
input ENUM_ALIGN_MODE   InpAlign=ALIGN_CENTER;       // Le moyen de l'alignement du tex
input bool              InpReadOnly=false;           // La possibilité d'éditer
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // L'angle du graphique pour le ra
input color             InpColor=clrBlack;           // La couleur du texte
```

© 2000-2015, MetaQuotes Software Corp.

```
input color            InpBackColor=clrWhite;       // La couleur du fond
input color            InpBorderColor=clrBlack;     // La couleur de la frontière
input bool             InpBack=false;               // L'objet à l'arrière-plan
input bool             InpSelection=false;          // Sélectionner pour les déplaceme
input bool             InpHidden=true;              // Est caché dans la liste des obj
input long             InpZOrder=0;                 // La priorité au clic d'une souri
//+------------------------------------------------------------------+
//| Crée l'objet "Le champ de saisie"                                |
//+------------------------------------------------------------------+
bool EditCreate(const long             chart_ID=0,               // ID du graphique
                const string           name="Edit",              // le nom de l'objet
                const int              sub_window=0,             // le numéro du sous-
                const int              x=0,                      // la coordonnée selo
                const int              y=0,                      // la coordonnée selo
                const int              width=50,                 // la largeur
                const int              height=18,                // la hauteur
                const string           text="Text",              // le texte
                const string           font="Arial",             // la fonte
                const int              font_size=10,             // la taille de la fo
                const ENUM_ALIGN_MODE  align=ALIGN_CENTER,       // le moyen de l'alig
                const bool             read_only=false,          // la possibilité d'é
                const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // l'angle du graphic
                const color            clr=clrBlack,             // la couleur du text
                const color            back_clr=clrWhite,        // la couleur du fond
                const color            border_clr=clrNONE,       // La couleur de la f
                const bool             back=false,               // à l'arrière-plan
                const bool             selection=false,          // Sélectionner pour
                const bool             hidden=true,              // est caché dans la
                const long             z_order=0)                // la priorité au cli
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons le champ de saisie"
   if(!ObjectCreate(chart_ID,name,OBJ_EDIT,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer l'objet \"Le champ de saisie\"! Le code de l'
      return(false);
     }
//---établissons les coordonnées de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- établissons la taille de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- établissons le texte
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- établissons la fonte du texte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- établissons la taille de la fonte
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//---  établissons le moyen de l'alignement du texte dans l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_ALIGN,align);
//--- activons (true) ou désactivons (false) le mode seulement pour la lecture
   ObjectSetInteger(chart_ID,name,OBJPROP_READONLY,read_only);
//--- établissons l'angle du graphique, relativement auquel les coordonnées de l'objet
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- établissons la couleur du texte
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons la couleur du fond
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- établissons la couleur de la frontière
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement de la marque par l
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace l'objet "Le champ de saisie"                             |
//+------------------------------------------------------------------+
bool EditMove(const long   chart_ID=0,  // ID du graphique
              const string name="Edit", // le nom de l'objet
              const int    x=0,         // la coordonnée selon l'axe X
              const int    y=0)         // la coordonnée selon l'axe Y
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée X de l'objet! Le code de l'e
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée Y de l'objet! Le code de l'e
      return(false);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change la taille de l'objet "Le champ de saisie"                 |
//+------------------------------------------------------------------+
bool EditChangeSize(const long   chart_ID=0,  // ID du graphique
                    const string name="Edit", // le nom de l'objet
                    const int    width=0,     // la largeur
                    const int    height=0)    // la hauteur
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons la taille de l'objet
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la largeur de l'objet! Le code de l'erreur
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le hauteur de l'objet! Le code de l'erreur
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le texte de l'objet "Le champ de saisie"                  |
//+------------------------------------------------------------------+
bool EditTextChange(const long   chart_ID=0,  // ID du graphique
                    const string name="Edit", // le nom de l'objet
                    const string text="Text") // le texte
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons le texte de l'objet
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le texte! Le code de l'erreur = ",GetLastEr
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Rend le texte de l'objet "Le champ de saisie"                          |
//+------------------------------------------------------------------------+
bool EditTextGet(string      &text,        // le texte
                 const long   chart_ID=0,  // ID du graphique
                 const string name="Edit") // le nom de l'objet
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons le texte de l'objet
   if(!ObjectGetString(chart_ID,name,OBJPROP_TEXT,0,text))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à recevoir le texte! Le code de l'erreur = ",GetLastE
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------+
//| Supprime l'objet "Le champ de saisie"                                  |
//+------------------------------------------------------------------------+
bool EditDelete(const long   chart_ID=0,  // ID du graphique
                const string name="Edit") // le nom de l'objet
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la marque
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer l'objet \"Le champ de saisie\"! Le code d
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------------+
//| Script program start function                                          |
//+------------------------------------------------------------------------+
void OnStart()
  {
//--- les tailles de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définirons les tailles de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("On n'a pas réussi à recevoir la largeur du graphique! Le code de l'erreur
      return;
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
      {
       Print("On n'a pas réussi à recevoir la hauteur du graphique! Le code de l'erreur
       return;
      }
//--- définirons le pas pour le changement de la taille du champ de saisie"
   int x_step=(int)x_distance/64;
//---établissons les coordonnées du champ et son taille
   int x=(int)x_distance/8;
   int y=(int)y_distance/2;
   int x_size=(int)x_distance/8;
   int y_size=InpFontSize*2;
//--- retiendrons le texte à la variable locale
   string text=InpText;
//--- créons le champ de saisie"
   if(!EditCreate(0,InpName,0,x,y,x_size,y_size,InpText,InpFont,InpFontSize,InpAlign,I
      InpCorner,InpColor,InpBackColor,InpBorderColor,InpBack,InpSelection,InpHidden,In
      {
       return;
      }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- agrandissons le champ de saisie
   while(x_size-x<x_distance*5/8)
      {
       //--- augmentons la largeur du champ de saisie
       x_size+=x_step;
       if(!EditChangeSize(0,InpName,x_size,y_size))
          return;
       //--- vérifions le fait de l'arrêt forcé du script
       if(IsStopped())
          return;
       //--- redessinons le graphique et attendons 0.05 seconde
       ChartRedraw();
       Sleep(50);
      }
//--- le retard à une demi-seconde
   Sleep(500);
//--- changeons le texte
   for(int i=0;i<20;i++)
      {
       //---ajoutons "+" au début et à la fin
       text="+"+text+"+";
       if(!EditTextChange(0,InpName,text))
          return;
       //--- vérifions le fait de l'arrêt forcé du script
       if(IsStopped())
```

© 2000-2015, MetaQuotes Software Corp.

```
         return;
       //--- redessinons le graphique et attendons 0.1 seconde
       ChartRedraw();
       Sleep(100);
      }
//--- le retard à une demi-seconde
   Sleep(500);
//--- supprimons le champ de saisie
   EditDelete(0,InpName);
   ChartRedraw();
//--- attendons 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_EVENT

L'objet "L'événement".

Note

Si vous passez votre souris sur l'événement, le texte apparaît.

L'exemple

Le script suivant crée et déplace l'objet "L'événement" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script construit l'objet graphique \"L'événement\"."
#property description "La date du point du rattachement est spécifiée en pourcentage d
#property description "la largeur de la fenêtre du graphique dans les barres."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string           InpName="Event";     // Le nom de l'événement
input int              InpDate=25;          // La date de l'événement en %
input string           InpText="Text";      // Le texte de l'événement
input color            InpColor=clrRed;     // La couleur de l'événement
input int              InpWidth=1;          // La taille du point à la sélection
input bool             InpBack=false;       // L'événement à l'arrière-plan
input bool             InpSelection=false; // Sélectionner pour les déplacements
input bool             InpHidden=true;      // Est caché dans la liste des objets
```

© 2000-2015, MetaQuotes Software Corp.

```
input long            InpZOrder=0;        // La priorité au clic d'une souris
//+------------------------------------------------------------------+
//| Crée l'objet "L'événement" sur le graphique                      |
//+------------------------------------------------------------------+
bool EventCreate(const long            chart_ID=0,       // ID du graphique
                 const string          name="Event",     // le nom de l'événement
                 const int             sub_window=0,     // le numéro du sous-fenêtre
                 const string          text="Text",      // le texte de l'événement
                 datetime              time=0,           // le temps
                 const color           clr=clrRed,       // la couleur
                 const int             width=1,          // l'épaisseur du point à la s
                 const bool            back=false,       // à l'arrière-plan
                 const bool            selection=false, // Sélectionner pour les dépla
                 const bool            hidden=true,      // est caché dans la liste des
                 const long            z_order=0)        // la priorité au clic d'une s
  {
//--- si le temps n'est pas spécifié, créons l'objet sur la dernière barre
   if(!time)
      time=TimeCurrent();
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons l'objet "L'événement".
   if(!ObjectCreate(chart_ID,name,OBJ_EVENT,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer \"L'événement\"! Le code de l'erreur = ",GetL
      return(false);
     }
//--- établissons le texte de l'événement
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- établissons la couleur
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons l'épaisseur du point du rattachement si l'objet est sélectionné
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement de l'événement par
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le texte de l'objet "L'événement"                         |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool EventTextChange(const long   chart_ID=0,   // ID du graphique
                     const string name="Event", // le nom de l'événement
                     const string text="Text")  // le texte
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons le texte de l'objet
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le texte! Le code de l'erreur = ",GetLastEr
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Le déplacement de l'objet "L'événement"                          |
//+------------------------------------------------------------------+
bool EventMove(const long   chart_ID=0,   // ID du graphique
               const string name="Event", // le nom de l'événement
               datetime     time=0)       // le temps
  {
//--- si le temps n'est pas spécifié, déplaçons l'événement sur la dernière barre
   if(!time)
      time=TimeCurrent();
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- déplaçons l'objet
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer l'objet \"L'événement\"! Le code de l'erre
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime l'objet "L'événement"                                   |
//+------------------------------------------------------------------+
bool EventDelete(const long   chart_ID=0,   // ID du graphique
                 const string name="Event") // le nom de l'événement
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons l'objet
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      Print(__FUNCTION__,
            ": on n'a pas réussi à supprimer \"L'événement\"! Le code de l'erreur = ",
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- vérifions les paramètres d'entrée à la validité
   if(InpDate<0 || InpDate>100)
     {
      Print("L'erreur! Les valeurs incorrectes des paramètres d'entrée!");
      return;
     }
//--- le nombre de barres visibles dans la fenêtre du graphique
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- le tableau pour stocker les valeurs des dates qui seront utilisées
//---pour l'établissement et le changement de la coordonnée du point du rattachement
   datetime date[];
//--- l'allocation de la mémoire
   ArrayResize(date,bars);
//--- remplissons le tableau des dates
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("On n'a pas réussi à copier les valeurs du temps! Le code de l'erreur = ",
      return;
     }
//--- définissons les points pour la création de l'objet
   int d=InpDate*(bars-1)/100;
//--- créons l'objet "L'événement".
   if(!EventCreate(0,InpName,0,InpText,date[d],InpColor,InpWidth,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- maintenant déplaçons l'objet
//--- le compteur du cycle
   int h_steps=bars/2;
//--- déplaçons l'objet
   for(int i=0;i<h_steps;i++)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- prenons la valeur suivante
      if(d<bars-1)
         d+=1;
      //--- deplaçons le point
      if(!EventMove(0,InpName,date[d]))
         return;
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //---redessinons le graphique
      ChartRedraw();
      // le retard à 0.05 seconde
      Sleep(50);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- supprimons le canal du graphique
   EventDelete(0,InpName);
   ChartRedraw();
//--- le retard à 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

OBJ_RECTANGLE_LABEL

L'objet "La marque rectangulaire".

Note

Les coordonnées des points du rattachement sont spécifiées en pixels. On peut choisir l'angle du rattachement de la marque rectangulaire de l'enumération ENUM_BASE_CORNER. On peut choisir le type du cadre pour la marque rectangulaire de l'enumérationENUM_BORDER_TYPE.

Est destinée à la création et la présentation de l'interface d'utilisateur graphique.

L'exemple

Le script suivant crée et déplace l'objet "La marque rectangulaire" sur le graphique. Pour la création et le changement des propriétés de l'objet graphique sont écrites les fonctions spéciales, lesquelles vous pouvez utiliser "comme il est" dans les programmes personnels.

```
//--- la description
#property description "Le script crée l'objet graphique \"La marque rectangulaire\"."
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée du script
input string           InpName="RectLabel";          // Le nom de la marque
input color            InpBackColor=clrSkyBlue;      // La couleur du fond
input ENUM_BORDER_TYPE InpBorder=BORDER_FLAT;        // Le type du cadre
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // L'angle du graphique pour le ra
input color            InpColor=clrDarkBlue;         // La couleur de la frontière plat
input ENUM_LINE_STYLE  InpStyle=STYLE_SOLID;         // Le style de la frontière plate
input int              InpLineWidth=3;               // L'épaisseur de la frontière pla
```

© 2000-2015, MetaQuotes Software Corp.

```
input bool              InpBack=false;              // L'objet à l'arrière-plan
input bool              InpSelection=true;          // Sélectionner pour les déplaceme
input bool              InpHidden=true;             // Est caché dans la liste des obj
input long              InpZOrder=0;                // La priorité au clic d'une souri
//+------------------------------------------------------------------+
//| Crée la marque rectangulaire                                     |
//+------------------------------------------------------------------+
bool RectLabelCreate(const long             chart_ID=0,               // ID du graphic
                     const string           name="RectLabel",         // le nom de la
                     const int              sub_window=0,             // le numéro du
                     const int              x=0,                      // la coordonnée
                     const int              y=0,                      // la coordonnée
                     const int              width=50,                 // la largeur
                     const int              height=18,                // la hauteur
                     const color            back_clr=C'236,233,216',  // la couleur du
                     const ENUM_BORDER_TYPE border=BORDER_SUNKEN,     // le type de la
                     const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // l'angle du gr
                     const color            clr=clrRed,               // la couleur de
                     const ENUM_LINE_STYLE  style=STYLE_SOLID,        // le style de l
                     const int              line_width=1,             // l'épaisseur d
                     const bool             back=false,               // à l'arrière-p
                     const bool             selection=false,          // sélectionner
                     const bool             hidden=true,              // est caché dan
                     const long             z_order=0)                // la priorité a
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- créons la marque rectangulaire
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à créer la marque rectangulaire! Le code de l'erreur
      return(false);
     }
//---  établissons les coordonnées de la marque
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- établissons la taille de la marque
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- établissons la couleur du fond
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- établissons le type de la frontière
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border);
//---établissons l'angle du graphique, relativement lequel les coordonnées du point se
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- établissons la couleur du cadre plat (en régime Flat)
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- établissons le style de la ligne du cadre plat
```

© 2000-2015, MetaQuotes Software Corp.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- établissons l'épaisseur de la frontière plate
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,line_width);
//--- affichons au premier-plan (false) ou à l'arrière-plan (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- activons (true) ou désactivons (false) le mode du déplacement de la marque par l
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- cachons (true) ou affichons (false) le nom de l'objet graphique dans la liste de
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//---établissons la priorité sur la réception de l'événement de la pression de la sour
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Déplace la marque rectangulaire                                  |
//+------------------------------------------------------------------+
bool RectLabelMove(const long   chart_ID=0,       // ID du graphique
                   const string name="RectLabel", // le nom de la marque
                   const int    x=0,              // la coordonnée selon l'axe X
                   const int    y=0)              // la coordonnée selon l'axe Y
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- deplaçons la marque rectangulaire
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée X de la marque! Le code de l
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à déplacer la coordonnée Y de la marque! Le code de l
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change la taille de la marque rectangulaire                      |
//+------------------------------------------------------------------+
bool RectLabelChangeSize(const long   chart_ID=0,       // ID du graphique
                         const string name="RectLabel", // le nom de la marque
                         const int    width=50,         // la largeur de la marque
                         const int    height=18)        // la hauteur de la marque
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons la taille de la marque
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer la largeur de la marque! Le code de l'erreu
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le hauteur de la marque! Le code de l'erreu
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Change le type de la frontière de la marque rectangulaire
//+------------------------------------------------------------------+
bool RectLabelChangeBorderType(const long             chart_ID=0,           // ID du g
                               const string           name="RectLabel",     // le nom
                               const ENUM_BORDER_TYPE border=BORDER_SUNKEN) // le type
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- changeons le type du cadre
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border))
     {
      Print(__FUNCTION__,
            ": on n'a pas réussi à changer le type de la frontière! Le code de l'erreu
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprime la marque rectangulaire                                 |
//+------------------------------------------------------------------+
bool RectLabelDelete(const long   chart_ID=0,       // ID du graphique
                     const string name="RectLabel") // le nom de la marque
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- supprimons la marque
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

© 2000-2015, MetaQuotes Software Corp.

```
            ": on n'a pas réussi à supprimer la marque rectangulaire! Le code de l'er
      return(false);
     }
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- les tailles de la fenêtre du graphique
   long x_distance;
   long y_distance;
//--- définirons les tailles de la fenêtre
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("On n'a pas réussi à recevoir la largeur du graphique! Le code de l'erreu
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("On n'a pas réussi à recevoir la hauteur du graphique! Le code de l'erreu
      return;
     }
//--- définirons les coordonnées de la marque rectangulaire
   int x=(int)x_distance/4;
   int y=(int)y_distance/4;
//--- établissons la taille de la marque
   int width=(int)x_distance/4;
   int height=(int)y_distance/4;
//--- créons la marque rectangulaire
   if(!RectLabelCreate(0,InpName,0,x,y,width,height,InpBackColor,InpBorder,InpCorner,
      InpColor,InpStyle,InpLineWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- changeons la taille de la marque rectangulaire
   int steps=(int)MathMin(x_distance/4,y_distance/4);
   for(int i=0;i<steps;i++)
     {
      //--- changeons la taille
      width+=1;
      height+=1;
      if(!RectLabelChangeSize(0,InpName,width,height))
         return;
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- vérifions le fait de l'arrêt forcé du script
      if(IsStopped())
         return;
      //--- redessinons le graphique et attendons 0.01 secondes
      ChartRedraw();
      Sleep(10);
     }
//--- le retard à 1 seconde
   Sleep(1000);
//--- changeons le type du cadre
   if(!RectLabelChangeBorderType(0,InpName,BORDER_RAISED))
      return;
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- changeons le type du cadre
   if(!RectLabelChangeBorderType(0,InpName,BORDER_SUNKEN))
      return;
//--- redessinons le graphique et attendons 1 seconde
   ChartRedraw();
   Sleep(1000);
//--- supprimons la marque
   RectLabelDelete(0,InpName);
   ChartRedraw();
//--- attendons 1 seconde
   Sleep(1000);
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

Les propriétés des objets

Chaque objet graphique sur le graphique des prix a un certain ensemble des propriétés. L'installation et la réception des valeurs des propriétés des objets est produite par les fonctions correspondantes du travail avec les objets graphiques. Pour chaque type de l'objet il y a un ensemble des propriétés, on énumère ici les valeurs toutes possibles de la famille des énumérations ENUM_OBJECT_PROPERTY. Certaines propriétés demandent la précision, comme par exemple, le numéro du niveau pour l'objet de l'extension de Fibonacci. Dans tels cas il est nécessaire d'indiquer la valeur du paramètre *modifier* dans les fonctions ObjectSet...() et ObjectGet...().

Les fonctions qui définissent les propriétés des objets graphiques, ainsi que les opérations de la création de ObjectCreate() et de déplacement ObjectMove() des objets sur le graphique, servent en réalité à envoyer des commandes au graphique. À l'exécution réussie de ces fonctions la commande se trouve dans une file d'attente des événements du graphique. Le changement visuel des propriétés des objets graphiques s'est produit en train du traitement de la file d'attente des événements du graphique donné.

Pour cette raison il ne faut pas attendre la mise à jour visuelle immédiate des objets graphiques après l'appel des fonctions données. En général, la mise à jour des objets graphiques s'est produite par le terminal automatiquement selon les événements du changement - l'entrée de la nouvelle cotation, le changement de la taille de la fenêtre du graphique etc.

Pour la mise à jour forcée des objets graphiques utilisez la commande pour redessiner le graphique ChartRedraw().

Pour les fonctions ObjectSetInteger() et ObjectGetInteger()

ENUM_OBJECT_PROPERTY_INTEGER

Identificateur	Description	Type de la propriété
OBJPROP_COLOR	La couleur	color
OBJPROP_STYLE	Le style	ENUM_LINE_STYLE
OBJPROP_WIDTH	L'épaisseur de la ligne	int
OBJPROP_BACK	L'objet à l'arrière-plan	bool
OBJPROP_ZORDER	La priorité de l'objet graphique sur la réception de l'événement de clic de la souris sur le graphique (CHARTEVENT_CLICK). Par défaut la valeur est égal au zéro à la création, mais la priorité peut être augmentée en cas de nécessité. En appliquant des objets un sur un autre, seulement un objet avec la plus haute priorité recevra l'événement CHARTEVENT_CLICK.	long

© 2000-2015, MetaQuotes Software Corp.

OBJPROP_FILL	Le remplissage d'un objet avec la couleur (pour OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)	bool
OBJPROP_HIDDEN	L'interdiction de montrer le nom d'un objet graphique dans la liste des objets du menu du terminal "Graphiques" - "Objets" - "Liste des objets". La valeur true permet de cacher l'objet inutile pour l'utilisateur de la liste. Par défaut true est défini pour les objets, qui affichent les événements du calendrier, l'histoire du commerce, ainsi que pour les objets créés du programme. Pour voir tels objets graphiques et recevoir l'accès à leurs propriétés, il faut appuyer sur le bouton "Tout" dans la fenêtre "Liste des objets".	bool
OBJPROP_SELECTED	L'objet est sélectionné	bool
OBJPROP_READONLY	La possibilité de l'édition du texte dans l'objet Edit	bool
OBJPROP_TYPE	Le type de l'objet	ENUM_OBJECT r/o
OBJPROP_TIME	La coordonnée du temps	datetime le modificateur=le numéro du point du rattachement
OBJPROP_SELECTABLE	L'accessibilité de l'objet	bool
OBJPROP_CREATETIME	Le temps de la création de l'objet	datetime r/o
OBJPROP_LEVELS	Le nombre de niveaux	int
OBJPROP_LEVELCOLOR	La couleur de la ligne-niveau	color le modificateur=le numéro du niveau
OBJPROP_LEVELSTYLE	Le style de la ligne-niveau	ENUM_LINE_STYLE le modificateur=le numéro du niveau
OBJPROP_LEVELWIDTH	L'épaisseur de la ligne-niveau	int le modificateur=le numéro du niveau

© 2000-2015, MetaQuotes Software Corp.

OBJPROP_ALIGN	L'alignement du texte à l'horizontale dans l'objet "Champ de l'entrée" (OBJ_EDIT)	ENUM_ALIGN_MODE
OBJPROP_FONTSIZE	La dimension de la fonte	int
OBJPROP_RAY_LEFT	Le rayon va à gauche	bool
OBJPROP_RAY_RIGHT	Le rayon va à droite	bool
OBJPROP_RAY	Une ligne verticale passe par toutes les fenêtres d'un graphique	bool
OBJPROP_ELLIPSE	L'affichage de l'ellipse complète pour l'objet "l'Arc de Fibonacci" (OBJ_FIBOARC)	bool
OBJPROP_ARROWCODE	Le code de la flèche pour l'objet "La flèche"	char
OBJPROP_TIMEFRAMES	La visibilité de l'objet sur les temps trames	le fonds des drapeaux flags
OBJPROP_ANCHOR	La position du point du rattachement de l'objet graphique	ENUM_ARROW_ANCHOR (pour OBJ_ARROW), ENUM_ANCHOR_POINT (pour OBJ_LABEL, OBJ_BITMAP_LABEL et OBJ_TEXT)
OBJPROP_XDISTANCE	La distance en pixels selon l'axe X de l'angle du rattachement (см. примечание)	int
OBJPROP_YDISTANCE	La distance en pixels selon l'axe Y de l'angle du rattachement (см. примечание)	int
OBJPROP_DIRECTION	La tendance de l'objet de Gann	ENUM_GANN_DIRECTION
OBJPROP_DEGREE	Le niveau du marquage d'onde d'Elliott	ENUM_ELLIOT_WAVE_DEGREE
OBJPROP_DRAWLINES	L'affichage des lignes pour le marquage d'onde d'Elliott	bool
OBJPROP_STATE	L'état du bouton (est appuyé/ est pressé)	bool
OBJPROP_CHART_ID	L'identificateur de l'objet "Graphique" (OBJ_CHART). Permet de travailler avec les propriétés de cet objet comme	long r/o

© 2000-2015, MetaQuotes Software Corp.

	avec le graphique ordinaire à l'aide des fonctions du paragraphe Les opérations avec les graphiques, mais il y a certains exceptions.	
OBJPROP_XSIZE	La largeur de l'objet selon l'axe X en pixels. Est spécifiée pour les objets OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	int
OBJPROP_YSIZE	La hauteur de l'objet selon l'axe Y en pixels. Est spécifiée pour les objets OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	int
OBJPROP_XOFFSET	La coordonnée X de l'angle supérieur gauche du domaine rectangulaire de la visibilité dans les objets graphiques "Marque graphique" et "Dessin" (OBJ_BITMAP_LABEL et OBJ_BITMAP). La valeur est spécifiée en pixels par rapport à l'angle supérieur gauche de l'image initiale.	int
OBJPROP_YOFFSET	La coordonnée Y de l'angle supérieur gauche du domaine rectangulaire de la visibilité dans les objets graphiques "Marque graphique" et "Dessin" (OBJ_BITMAP_LABEL et OBJ_BITMAP). La valeur est spécifiée en pixels par rapport à l'angle supérieur gauche de l'image initiale.	int
OBJPROP_PERIOD	La période pour l'objet "le graphique"	ENUM_TIMEFRAMES
OBJPROP_DATE_SCALE	Affichage de l'échelle du temps pour l'objet "Graphique"	bool
OBJPROP_PRICE_SCALE	Affichage de l'échelle de prix pour l'objet "Graphique"	bool

© 2000-2015, MetaQuotes Software Corp.

OBJPROP_CHART_SCALE	L'échelle pour l'objet "le graphique"	int la valeur dans le domaine 0-5
OBJPROP_BGCOLOR	La couleur du fond pour OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL	color
OBJPROP_CORNER	L'angle du graphique pour le rattachement de l'objet graphique	ENUM_BASE_CORNER
OBJPROP_BORDER_TYPE	Le type du cadre pour l'objet "Le cadre rectangulaire"	ENUM_BORDER_TYPE
OBJPROP_BORDER_COLOR	La couleur du cadre pour l'objet OBJ_EDIT et OBJ_BUTTON	color

A l'application des opérations avec les graphiques pour l'objet "Graphique" (OBJ_CHART) il y a les limitations suivantes

- ne peut être fermé à l'aide de ChartClose();
- on ne peut pas changer le symbole/période à l'aide de la fonction ChartSetSymbolPeriod();
- Les propriétés CHART_SCALE, CHART_BRING_TO_TOP, CHART_SHOW_DATE_SCALE et CHART_SHOW_PRICE_SCALE (ENUM_CHART_PROPERTY_INTEGER) ne fonctionnent pas.

Un mode spécial d'affichage de l'image peut être défini par le programmepour les objets OBJ_BITMAP_LABEL et OBJ_BITMAP . Dans ce mode, seulement une partie d'une image initiale, à laquelle un domaine rectangulaire visible est appliquée, est montrée tandis que le reste de l'image devient invisible. Il est nécessaire de définir les dimensions du domaine de la visibilité à l'aide des propriétés OBJPROP_XSIZE et OBJPROP_YSIZE. On peut "déplacer" le domaine de la visibilité seulement dans la limite de l'image initiale à l'aide des propriétés OBJPROP_XOFFSET et OBJPROP_YOFFSET.

Для объектов с фиксированными размерами: OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT и OBJ_CHART свойства OBJPROP_XDISTANCE и OBJPROP_YDISTANCE задают положение левой верхней точки объекта относительно угла графика (OBJPROP_CORNER), от которого будут отсчитываться координаты X и Y в пикселях.

Pour les fonctions ObjectSetDouble() et ObjectGetDouble()

ENUM_OBJECT_PROPERTY_DOUBLE

Identificateur	Description	Type de la propriété
OBJPROP_PRICE	La coordonnée du prix	double le modificateur=le numéro du point du rattachement
OBJPROP_LEVELVALUE	La valeur du niveau	double le modificateur=le

© 2000-2015, MetaQuotes Software Corp.

		numéro du niveau
OBJPROP_SCALE	L'échelle (la propriété des objets de Gann et l'objet "les arcs de Fibonacci")	double
OBJPROP_ANGLE	L'angle. Pour les objets créés du programme avec l'angle qui ne pas encore spécifié, la valeur est égale à EMPTY_VALUE	double
OBJPROP_DEVIATION	La déviation pour le canal de la déviation standard	double

Pour les fonctions ObjectSetString() et ObjectGetString()

ENUM_OBJECT_PROPERTY_STRING

Identificateur	Description	Type de la propriété
OBJPROP_NAME	Le nom de l'objet	string
OBJPROP_TEXT	La description de l'objet (le texte qui se trouve dans l'objet)	string
OBJPROP_TOOLTIP	Le texte de l'infobulle émergeante. Si la propriété n'est pas spécifiée, se montre l'infobulle automatiquement formée par le terminal. On peut désactiver l'infobulle, en définissant sa valeur "\n" (saut de la ligne)	string
OBJPROP_LEVELTEXT	La description du niveau	string le modificateur=le numéro du niveau
OBJPROP_FONT	La fonte	string
OBJPROP_BMPFILE	Le nom du fichier BMP pour l'objet "Une balise graphique". Voir aussi Ressources	string le modificateur: 0-l'état ON, 1-l'état OFF
OBJPROP_SYMBOL	Le caractère pour l'objet "le graphique"	string

Pour l'objetOBJ_RECTANGLE_LABEL ("Marque rectangulaire") on peut spécifier l'un des trois modes d'affichage, auxquels correspondent les valeurs de l'énumération ENUM_BORDER_TYPE.

ENUM_BORDER_TYPE

© 2000-2015, MetaQuotes Software Corp.

Identificateur	La description
BORDER_FLAT	Forme plate
BORDER_RAISED	Forme proéminente
BORDER_SUNKEN	Forme concave

Pour l'objet OBJ_EDIT ("Champ de l'entrée") et pour la fonction ChartScreenShot() on peut indiquer le type de l'alignement à l'horizontale à l'aide des valeurs de l'énumération ENUM_ALIGN_MODE.

ENUM_ALIGN_MODE

Identificateur	La description
ALIGN_LEFT	L'alignement selon une frontière gauche
ALIGN_CENTER	L'alignement selon le centre (seulement pourl'objet "Champ de l'entrée")
ALIGN_RIGHT	L'alignement selon une frontière droite

Exemple:

```
#define  UP          "\x0431"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="my_OBJ_LABEL_object";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Object ",label_name," not found. Error code = ",GetLastError());
      //--- créons l'objet Label
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- établissons la coordonnée X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- établissons la coordonnée Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      //--- spécifions la couleur du texte
      ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrWhite);
      //--- établissons le texte pour l'objet Label
      ObjectSetString(0,label_name,OBJPROP_TEXT,UP);
      //--- établissons la fonte de l'inscription
      ObjectSetString(0,label_name,OBJPROP_FONT,"Wingdings");
      //--- définissons la taille de la fonte
      ObjectSetInteger(0,label_name,OBJPROP_FONTSIZE,10);
      //--- tournerons sur 45 degrés selon l'aiguille d'une montre
      ObjectSetDouble(0,label_name,OBJPROP_ANGLE,-45);
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- interdisons la sélection de l'objet par la souris
      ObjectSetInteger(0,label_name,OBJPROP_SELECTABLE,false);
      //--- dessinerons sur le graphique
      ChartRedraw(0);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Les méthodes du rattachement des objets

Les objets graphiques Text et Label (OBJ_TEXT, OBJ_BITMAP_LABEL et OBJ_LABEL) peuvent avoir un de 9 divers moyens du rattachement des coordonnées. On peut indiquer une variante nécessaire à l'aide de la fonction ObjectSetInteger (handle_du graphique, le nom_de l'objet, OBJPROP_ANCHOR, la mode_du rattachement), où la mode_du rattachement - une des valuers de l'énumération ENUM_ANCHOR_POINT.

ENUM_ANCHOR_POINT

Identificateur	Description
ANCHOR_LEFT_UPPER	Le point du rattachement dans un angle gauche supérieur
ANCHOR_LEFT	Le point du rattachement à gauche selon le centre
ANCHOR_LEFT_LOWER	Le point du rattachement dans un angle gauche inférieur
ANCHOR_LOWER	Le point du rattachement d'en bas selon le centre
ANCHOR_RIGHT_LOWER	Le point du rattachement dans un angle droit inférieur
ANCHOR_RIGHT	Le point du rattachement à droite selon le centre
ANCHOR_RIGHT_UPPER	Le point du rattachement à droite selon le centre
ANCHOR_UPPER	Le point du rattachement par dessus selon le centre
ANCHOR_CENTER	Le point du rattachement strictement selon le centre de l'objet

Объекты OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT и OBJ_CHART имеют фиксированную точку привязки в левом верхнем углу (ANCHOR_LEFT_UPPER).

Exemple:

```
string text_name="my_OBJ_TEXT_object";
if(ObjectFind(0,text_name)<0)
  {
   Print("Object ",text_name," not found. Error code = ",GetLastError());
   //--- recevrons le prix maximum du graphique
   double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
   //--- créons l'objet Label
   ObjectCreate(0,text_name,OBJ_TEXT,0,TimeCurrent(),chart_max_price);
   //--- spécifions la couleur du texte
   ObjectSetInteger(0,text_name,OBJPROP_COLOR,clrWhite);
   //--- spécifions la couleur du fond
```

© 2000-2015, MetaQuotes Software Corp.

```
      ObjectSetInteger(0,text_name,OBJPROP_BGCOLOR,clrGreen);
      //--- établissons le texte pour l'objet Label
      ObjectSetString(0,text_name,OBJPROP_TEXT,TimeToString(TimeCurrent()));
      //--- établissons la fonte de l'inscription
      ObjectSetString(0,text_name,OBJPROP_FONT,"Trebuchet MS");
      //--- établissons la dimension de la fonte
      ObjectSetInteger(0,text_name,OBJPROP_FONTSIZE,10);
      //--- établissons le rattachement à l'angle droit supérieur
      ObjectSetInteger(0,text_name,OBJPROP_ANCHOR,ANCHOR_RIGHT_UPPER);
      //--- tournerons sur 90 degrés contre l'aiguille d'une montre
      ObjectSetDouble(0,text_name,OBJPROP_ANGLE,90);
      //--- interdisons la sélection de l'objet par la souris
      ObjectSetInteger(0,text_name,OBJPROP_SELECTABLE,false);
      //--- dessinerons sur le graphique
      ChartRedraw(0);
     }
```

Les objets graphiques Arrow (OBJ_ARROW) ont seulement 2 moyens du rattachement de ses coordonnées. Les identificateurs sont énumérés dans ENUM_ARROW_ANCHOR.

ENUM_ARROW_ANCHOR

Identificateur	Description
ANCHOR_TOP	Le point du rattachement pour la flèche se trouve par dessus
ANCHOR_BOTTOM	Le point du rattachement pour la flèche se trouve en bas

Exemple:

```
void OnStart()
  {
//--- tableaux de service
   double Ups[],Downs[];
   datetime Time[];
//--- définissons les tableaux comme la série temporelle
   ArraySetAsSeries(Ups,true);
   ArraySetAsSeries(Downs,true);
   ArraySetAsSeries(Time,true);
//--- créons le handle de l'indicateur Fractals
   int FractalsHandle=iFractals(NULL,0);
   Print("FractalsHandle =",FractalsHandle);
//--- enlevons le code de l'erreur
   ResetLastError();
//--- essayons de copier les valeurs de l'indicateur
   int copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Print("On n'a pas réussi à copier les fractals supérieurs. Error = ",GetLastErro
```

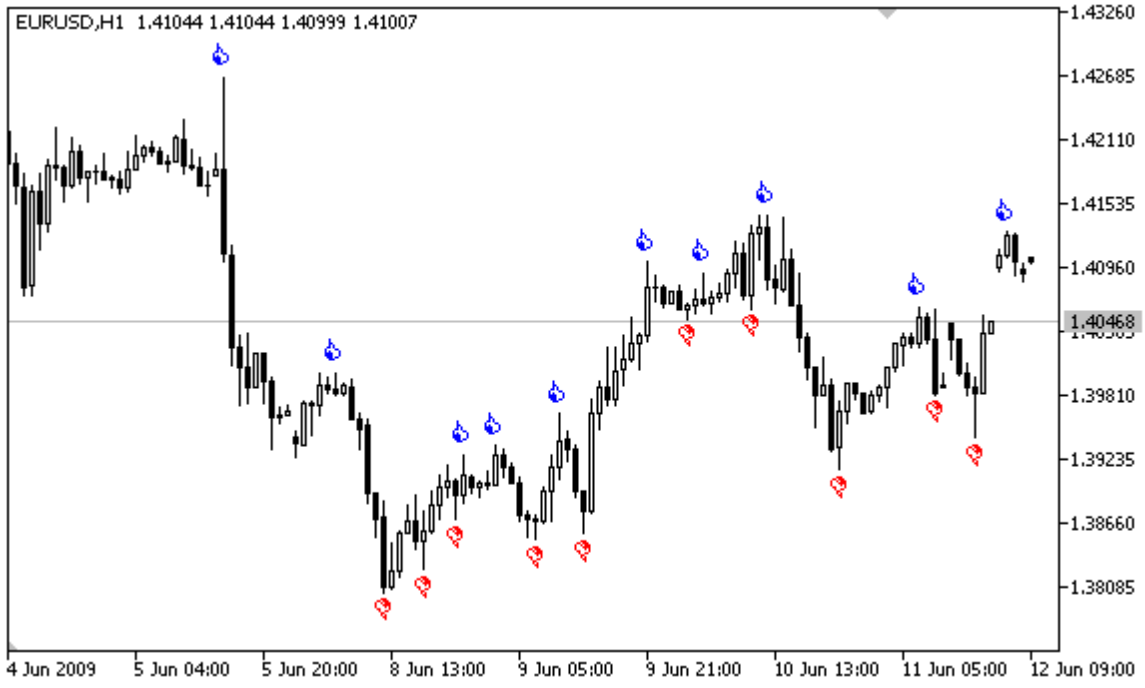
© 2000-2015, MetaQuotes Software Corp.

```
      return;
     }

   ResetLastError();
//--- essayons de copier les valeurs de l'indicateur
   copied=CopyBuffer(FractalsHandle,1,0,1000,Downs);
   if(copied<=0)
     {
      Print("On n'a pas réussi à copier les fractals inférieurs. Error = ",GetLastErro
      return;
     }

   ResetLastError();
//--- copions la série temporelle, contenante le temps de l'ouverture des dernières 10
   copied=CopyTime(NULL,0,0,1000,Time);
   if(copied<=0)
     {
      Print("On n'a pas réussi à copier les temps de l'ouverture pour les dernières 10
      return;
     }

   int upcounter=0,downcounter=0; // comptons là bas le nombre de flèches
   bool created;// recevrons le résultat de la tentative de la création de l'objet
   for(int i=2;i<copied;i++)// Parcourons les valeurs de l'indicateur iFractals
     {
      if(Ups[i]!=EMPTY_VALUE)// Ont trouvé un fractal supérieur
        {
         if(upcounter<10)// créons pas plus de 10 objets "en haut"
           {
            //--- essayons de créer l'objet "en haut"
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_UP,0,Time[i],Ups[i]
            if(created)// S'il a été créé - nous faisons à lui le restylage
              {
               //--- le point du rattachement est ci-dessous pour ne pas couvrir la ba
               ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_BOTTOM);
               //--- étape finale - peindre
               ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrBlue);
               upcounter++;
              }
           }
        }
      if(Downs[i]!=EMPTY_VALUE)// ont trouvé un fractal inférieur
        {
         if(downcounter<10)// créons pas plus de 10 objets "en bas"
           {
            //--- essayons de créer l'objet "en bas"
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_DOWN,0,Time[i],Dowr
            if(created)//s'il a été créé - nous faisons à lui le restylage
              {
```

© 2000-2015, MetaQuotes Software Corp.

```
                //--- point du rattachement par dessus pour ne pas couvrir la barre
                ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_TOP);
                //--- étape finale - peindre
                ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrRed);
                downcounter++;
               }
            }
         }
      }
   }
```

Après l'exécution de script le graphique ressemblera sur ce dessin.



© 2000-2015, MetaQuotes Software Corp.

L'angle du graphique, à quel un objet est attaché

Il y a une série des objets graphiques, pour lesquelles on peut donner l'angle du graphique, par rapport auquel on indique les coordonnées en pixels. Ce sont les types suivants des objets (les identificateurs du type de l'objet sont indiqués entre les parenthèses):

- Label (OBJ_LABEL);
- Button (OBJ_BUTTON);
- Chart (OBJ_CHART);
- Bitmap Label (OBJ_BITMAP_LABEL);
  - Rectangle Label (OBJ_RECTANGLE_LABEL);
- Edit (OBJ_EDIT).

Pour indiquer l'angle du graphique, du quel on comptera les coordonnées X et Y en pixels, il est nécessaire de se servir de la fonction ObjectSetInteger(chartID, name, OBJPROP_CORNER, **chart_corner**), où:

- chartID - l'identificateur du graphique;
- name – le nom de l'objet graphique;
- OBJPROP_CORNER – l'identificateur de la propriété pour spécifier l'angle du rattachement;
- **chart_corner** – l'angle demandé du graphique, peut accepter une des valeurs de l'énumération ENUM_BASE_CORNER.

ENUM_BASE_CORNER

Identificateur	Description
CORNER_LEFT_UPPER	Le centre des coordonnées dans un angle gauche supérieur du graphique
CORNER_LEFT_LOWER	Le centre des coordonnées dans un angle gauche inférieur du graphique
CORNER_RIGHT_LOWER	Le centre des coordonnées dans un angle droit inférieur du graphique
CORNER_RIGHT_UPPER	Le centre des coordonnées dans un angle droit supérieur du graphique

Exemple:

```
void CreateLabel(long   chart_id,
                 string name,
                 int    chart_corner,
                 string text_label,
                 int    x_ord,
                 int    y_ord)
  {
//---
   ObjectCreate(chart_id,name,OBJ_LABEL,0,0,0);
   ResetLastError();
   if(!ObjectSetInteger(chart_id,name,OBJPROP_CORNER,chart_corner))
```

© 2000-2015, MetaQuotes Software Corp.

```
      Print("On n'a pas réussi à établir l'angle du rattachement pour l'objet ",
            name,", une erreur ",GetLastError());
   ObjectSetInteger(chart_id,name,OBJPROP_XDISTANCE,x_ord);
   ObjectSetInteger(chart_id,name,OBJPROP_YDISTANCE,y_ord);
   ObjectSetString(chart_id,name,OBJPROP_TEXT,text_label);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   string arrows[4]={"LEFT_UPPER","RIGHT_UPPER","RIGHT_LOWER","LEFT_LOWER"};
   CreateLabel(0,arrows[0],CORNER_LEFT_UPPER,"0",50,50);
   CreateLabel(0,arrows[1],CORNER_RIGHT_UPPER,"1",50,50);
   CreateLabel(0,arrows[2],CORNER_RIGHT_LOWER,"2",50,50);
   CreateLabel(0,arrows[3],CORNER_LEFT_LOWER,"3",50,50);
  }
```

© 2000-2015, MetaQuotes Software Corp.

La visibilité des objets

La combinaison des drapeaux de la visibilité de l'objet définit les temps trames du graphique, où l'objet est affiché. Pour l'installation/réception de la valeur de la propriété OBJPROP_TIMEFRAMES on peut utiliser les fonctions ObjectSetInteger()/ObjectGetInteger().

Constante	Valeur	Description
OBJ_PERIOD_M1	0x00000001	L'objet se dessine sur les graphiques de 1 minutes
OBJ_PERIOD_M2	0x00000002	L'objet se dessine sur les graphiques de 2 minutes
OBJ_PERIOD_M3	0x00000004	L'objet se dessine sur les graphiques de 3 minutes
OBJ_PERIOD_M4	0x00000008	L'objet se dessine sur les graphiques de 4 minutes
OBJ_PERIOD_M5	0x00000010	L'objet se dessine sur les graphiques de 5 minutes
OBJ_PERIOD_M6	0x00000020	L'objet se dessine sur les graphiques de 6 minutes
OBJ_PERIOD_M10	0x00000040	L'objet se dessine sur les graphiques de 10 minutes
OBJ_PERIOD_M12	0x00000080	L'objet se dessine sur les graphiques de 12 minutes
OBJ_PERIOD_M15	0x00000100	L'objet se dessine sur les graphiques de 15 minutes
OBJ_PERIOD_M20	0x00000200	L'objet se dessine sur les graphiques de 20 minutes
OBJ_PERIOD_M30	0x00000400	L'objet se dessine sur les graphiques de 30 minutes
OBJ_PERIOD_H1	0x00000800	L'objet se dessine sur les graphiques de 1 heure
OBJ_PERIOD_H2	0x00001000	L'objet se dessine sur les graphiques de 2 heures
OBJ_PERIOD_H3	0x00002000	L'objet se dessine sur les graphiques de 3 heures
OBJ_PERIOD_H4	0x00004000	L'objet se dessine sur les graphiques de 4 heures
OBJ_PERIOD_H6	0x00008000	L'objet se dessine sur les graphiques de 6 heures
OBJ_PERIOD_H8	0x00010000	L'objet se dessine sur les graphiques de 8 heures

© 2000-2015, MetaQuotes Software Corp.

OBJ_PERIOD_H12	0x00020000	L'objet se dessine sur les graphiques de 12 heures
OBJ_PERIOD_D1	0x00040000	L'objet se dessine sur les graphiques de jour
OBJ_PERIOD_W1	0x00080000	L'objet se dessine sur les graphiques d'une semaine
OBJ_PERIOD_MN1	0x00100000	L'objet se dessine sur les graphiques des mois
OBJ_ALL_PERIODS	0x001fffff	L'objet se dessine sur tous les temps trames

On peut combiner les drapeaux de la visibilité avec l'aide du caractère "|", par exemple, la combinaison des drapeaux OBJ_PERIOD_M10|OBJ_PERIOD_H4 signifie que l'objet sera visible sur les temps trames de 10 minutes et de 4 heures.

Exemple:

```
void OnStart()
  {
//---
   string highlevel="PreviousDayHigh";
   string lowlevel="PreviousDayLow";
   double prevHigh;           // High du jour précédent
   double prevLow;            // Low du jour précédent
   double highs[],lows[];     // les tableaux pour la réception High et Low

//--- réinitialisez une dernière erreur
   ResetLastError();
//--- recevrons les 2 dernières significations High sur le temps trame du jour
   int highsgot=CopyHigh(Symbol(),PERIOD_D1,0,2,highs);
   if(highsgot>0) // si le copiage a passé avec succès
     {
      Print("Les prix High pour les derniers 2 jours sont reçus avec succès ");
      prevHigh=highs[0]; // High du jour précédent
      Print("prevHigh = ",prevHigh);
      if(ObjectFind(0,highlevel)<0) // L'objet avec le nom highlevel n'est pas trouvé
        {
         ObjectCreate(0,highlevel,OBJ_HLINE,0,0,0); // créons l'objet la ligne horizor
        }
      //--- spécifions le niveau de prix pour la ligne highlevel
      ObjectSetDouble(0,highlevel,OBJPROP_PRICE,0,prevHigh);
      //--- établissons la visibilité seulement pour PERIOD_M10 et PERIOD_H4
      ObjectSetInteger(0,highlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else
     {
      Print("On n'a pas réussi à recevoir les prix High pour les derniers 2 jours, Er
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- réinitialisez une dernière erreur
   ResetLastError();
//--- recevrons les 2 dernières significations Low sur le temps trame du jour
   int lowsgot=CopyLow(Symbol(),PERIOD_D1,0,2,lows);
   if(lowsgot>0) // si le copiage a passé avec succès
     {
      Print("Les prix Low pour les derniers 2 jours sont reçus avec succès");
      prevLow=lows[0]; // Low du jour précédent
      Print("prevLow =",prevLow);
      if(ObjectFind(0,lowlevel)<0) // l'objet avec le nom lowlevel n'est pas trouvé
        {
         ObjectCreate(0,lowlevel,OBJ_HLINE,0,0,0); // créons l'objet la ligne horizont
        }
      //--- spécifions le niveau de prix pour la ligne lowlevel
      ObjectSetDouble(0,lowlevel,OBJPROP_PRICE,0,prevLow);
      //--- établissons la visibilité seulement pour  PERIOD_M10 et PERIOD_H4
      ObjectSetInteger(0,lowlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else Print("On n'a pas réussi à recevoir les prix Low pour les derniers 2 jours, Er

   ChartRedraw(0); // redessinerons le graphique forcément
  }
```

Voir aussi

PeriodSeconds, Period, Les périodes des graphiques, Date et temps

© 2000-2015, MetaQuotes Software Corp.

Les niveaux des ondes d'Elliott

Les ondes d'Elliott sont réprésentées par deux objets graphiques des types OBJ_ELLIOTWAVE5 et OBJ_ELLIOTWAVE3. Pour spécifier la dimension de l'onde (le moyen de l'étiquetage des ondes) la propriété OBJPROP_DEGREE est utilisée, auquel on peut établir une des valeurs d'énumération ENUM_ELLIOT_WAVE_DEGREE.

ENUM_ELLIOT_WAVE_DEGREE

Constante	Description
ELLIOTT_GRAND_SUPERCYCLE	Un Supercycle principal (Grand Supercycle)
ELLIOTT_SUPERCYCLE	Un Supercycle (Supercycle)
ELLIOTT_CYCLE	Une boucle (Cycle)
ELLIOTT_PRIMARY	Une boucle primaire (Primary)
ELLIOTT_INTERMEDIATE	Un chaînon intermédiaire (Intermediate)
ELLIOTT_MINOR	Une boucle secondaire (Minor)
ELLIOTT_MINUTE	Une minute (Minute)
ELLIOTT_MINUETTE	Un seconde (Minuette)
ELLIOTT_SUBMINUETTE	Un subseconde (Subminuette)

Exemple:

```
   for(int i=0;i<ObjectsTotal(0);i++)
     {
      string currobj=ObjectName(0,i);
      if((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE3) ||
         ((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE5)))
        {
         //--- établissons le niveau du marquage dans INTERMEDIATE
         ObjectSetInteger(0,currobj,OBJPROP_DEGREE,ELLIOTT_INTERMEDIATE);
         //--- branchons la projection des lignes entre les tops des ondes
         ObjectSetInteger(0,currobj,OBJPROP_DRAWLINES,true);
         //--- établissons la couleur des lignes
         ObjectSetInteger(0,currobj,OBJPROP_COLOR,clrBlue);
         //--- établissons l'épaisseur des lignes
         ObjectSetInteger(0,currobj,OBJPROP_WIDTH,5);
         //--- déterminons la description
         ObjectSetString(0,currobj,OBJPROP_TEXT,"test script");
        }
     }
```

© 2000-2015, MetaQuotes Software Corp.

Les objets de Gann

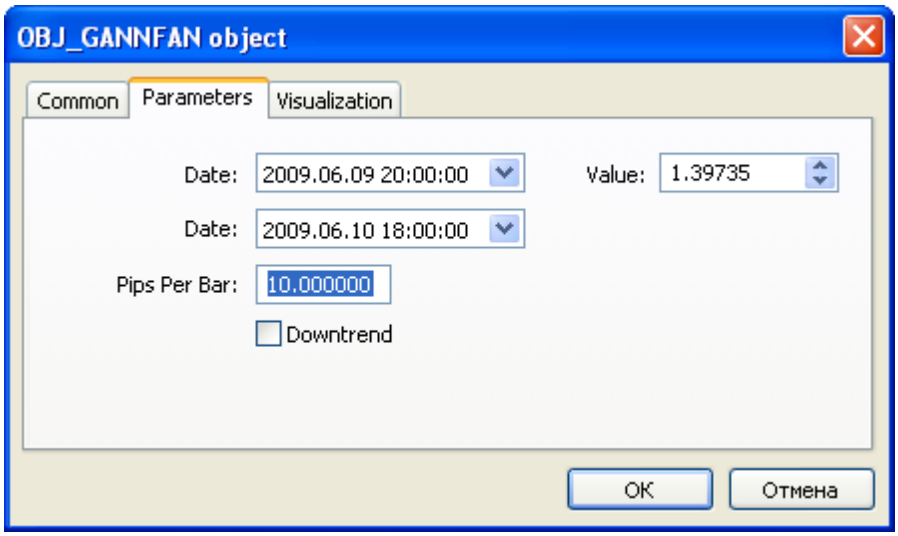
Pour les objets du fanion de Gann (OBJ_GANNFAN) et la grille de Gann (OBJ_GANNGRID) on peut indiquer une de deux valeurs de l'énumération ENUM_GANN_DIRECTION, qui détermine la direction de la tendance.

ENUM_GANN_DIRECTION

Constante	Description
GANN_UP_TREND	La ligne correspond à la tendance montante
GANN_DOWN_TREND	La ligne correspond à la tendance descendante

Pour l'installation de l'échelle de la ligne principale 1x1 on utilise la fonction ObjectSetDouble (chart_handle, gann_object_name, OBJPROP_SCALE, scale), où:

- chart_handle - la fenêtre du graphique, où se trouve un objet;
- gann_object_name - le nom de l'objet;
- OBJPROP_SCALE - l'identificateur de la propriété "Scale";
- scale - l'échelle demandée dans les unités Pips/Bar.



L'exemple de la création du fanion de Gann:

```
void OnStart()
  {
//---
   string my_gann="OBJ_GANNFAN object";
   if(ObjectFind(0,my_gann)<0)// objet n'est pas trouvé
     {
      //--- informons de l'échec
      Print("Object ",my_gann," not found. Error code = ",GetLastError());
      //--- recevrons un prix maximum du graphique
      double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
      //--- recevrons un prix minimum du graphique
      double chart_min_price=ChartGetDouble(0,CHART_PRICE_MIN,0);
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- combien de barres sont montrées sur le graphique?
      int bars_on_chart=ChartGetInteger(0,CHART_VISIBLE_BARS);
      //--- créons le tableau, où nous inscrirons le temps de l'ouverture de chaque ba
      datetime Time[];
      //--- organiserons l'accès au tableau comme dans la série temporelle
      ArraySetAsSeries(Time,true);
      //--- maintenant copions dedans les données des barres visibles sur le graphique
      int times=CopyTime(NULL,0,0,bars_on_chart,Time);
      if(times<=0)
        {
         Print("On n'a pas réussi à copier le tableau avec le temps d'ouverture!");
         return;
        }
      //--- les préparations préalables sont finies

      //--- index de la barre centrale sur le graphique
      int center_bar=bars_on_chart/2;
      //--- équateur du graphique est entre le maximum et le minimum
      double mean=(chart_max_price+chart_min_price)/2.0;
      //--- établissons les coordonnées du premier point du rattachement au centre
      ObjectCreate(0,my_gann,OBJ_GANNFAN,0,Time[center_bar],mean,
                   //--- le deuxième point du rattachement est à droite
                   Time[center_bar/2],(mean+chart_min_price)/2.0);
      Print("Time[center_bar] = "+(string)Time[center_bar]+"  Time[center_bar/2] = "+
      //Print("Time[center_bar] = Time[center_bar/2]);
      //--- établissons l'échelle dans les unités Pips/Bar
      ObjectSetDouble(0,my_gann,OBJPROP_SCALE,10);
      //--- établissons l'épaisseur des lignes
      ObjectSetInteger(0,my_gann,OBJPROP_DIRECTION,GANN_UP_TREND);
      //--- établissons l'épaisseur des lignes
      ObjectSetInteger(0,my_gann,OBJPROP_WIDTH,1);
      //--- déterminé le style des lignes
      ObjectSetInteger(0,my_gann,OBJPROP_STYLE,STYLE_DASHDOT);
      //--- et la couleur des lignes
      ObjectSetInteger(0,my_gann,OBJPROP_COLOR,clrYellowGreen);
      //--- permettons à l'utilisateur de sélectionner l'objet
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTABLE,true);
      //--- sélectionnons-le
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTED,true);
      //--- dessinons-le sur le graphique
      ChartRedraw(0);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

L'ensembles des couleurs Web

On définit les constantes suivantes colorées pour le type color:

						clrDarkTurquoise	
clrLightSeaGreen							
clrGoldenrod	clrMediumSpringGreen	clrLawnGreen					
	clrOrange	clrGold	clrYellow	clrChartreuse	clrLime	clrSpringGreen	clrAqua
clrDeepSkyBlue		clrMagenta					
		clrMediumTurquoise		clrTurquoise			clrDarkKhaki
		clrGreenYellow	clrMediumAquamarine				clrOrchid
clrMediumPurple				clrDarkGray	clrSandyBrown		clrTan
	clrBurlyWood	clrHotPink		clrViolet		clrSkyBlue	clrLightSalmon
clrPlum	clrKhaki	clrLightGreen	clrAquamarine	clrSilver	clrLightSkyBlue	clrLightSteelBlue	clrLightBlue
clrPaleGreen	clrThistle	clrPowderBlue	clrPaleGoldenrod	clrPaleTurquoise	clrLightGray	clrWheat	clrNavajoWhite
clrMoccasin	clrLightPink	clrGainsboro	clrPeachPuff	clrPink	clrBisque	clrLightGoldenrod	clrBlanchedAlmond
clrLemonChiffon	clrBeige	clrAntiqueWhite	clrPapayaWhip	clrCornsilk	clrLightYellow	clrLightCyan	clrLinen
clrLavender	clrMistyRose	clrOldLace	clrWhiteSmoke	clrSeashell	clrIvory	clrHoneydew	clrAliceBlue
clrLavenderBlush	clrMintCream	clrSnow	clrWhite				

On peut spécifier la couleur pour les objets à l'aide de la fonction ObjectSetInteger() et pour les indicateurs d'utilisateur à l'aide de la fonction PlotIndexSetInteger(). Pour recevoir des valeurs de couleur il y a des fonctions analogiques ObjectGetInteger() et PlotIndexGetInteger().

© 2000-2015, MetaQuotes Software Corp.

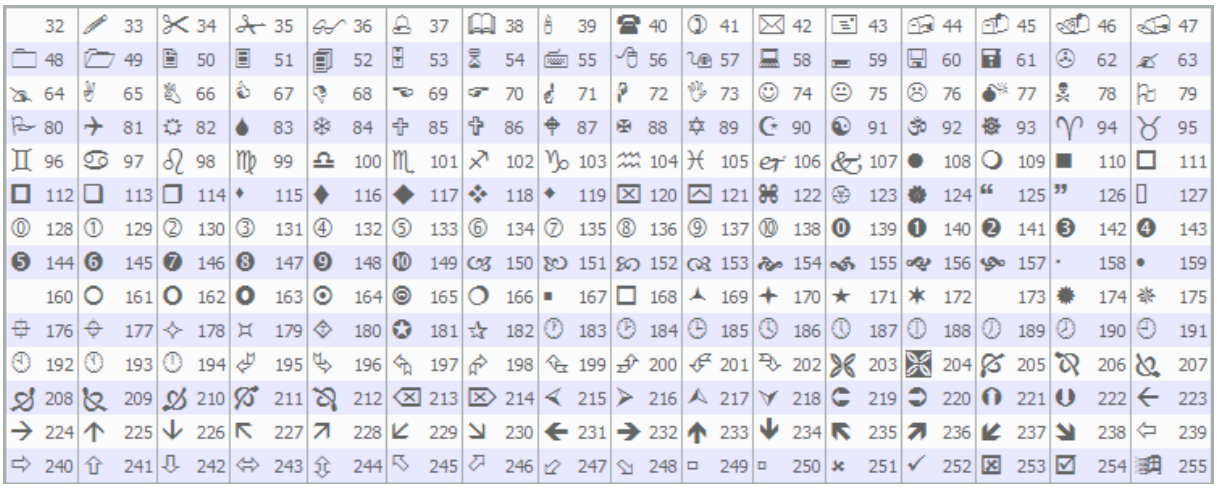
Exemple:

```
//---- indicator settings
#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_type3   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_color2  clrRed
#property indicator_color3  clrLime
```

© 2000-2015, MetaQuotes Software Corp.

Wingdings

Les caractères de la fonte Wingdings, utilisés avec l'objet OBJ_ARROW:



L'installation du caractère nécessaire est produite à l'aide de la fonction ObjectSetInteger().

Exemple:

```
void OnStart()
  {
//---
   string up_arrow="up_arrow";
   datetime time=TimeCurrent();
   double lastClose[1];
   int close=CopyClose(Symbol(),Period(),0,1,lastClose);        // recevrons le prix C
//--- si le prix est reçu
   if(close>0)
     {
      ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);           // créons la flèche
      ObjectSetInteger(0,up_arrow,OBJPROP_ARROWCODE,241);     // établissons le code de
      ObjectSetInteger(0,up_arrow,OBJPROP_TIME,time);         // déterminons le temps
      ObjectSetDouble(0,up_arrow,OBJPROP_PRICE,lastClose[0]);// déterminons le prix
      ChartRedraw(0);                                         // copions la fenêtre
     }
   else
      Print("On n'a pas réussi à recevoir le dernier prix Close!");
  }
```

© 2000-2015, MetaQuotes Software Corp.

Les constantes des indicateurs

Les 37 indicateurs techniques standards sont prédéterminés, qui peuvent être utilisés dans les programmes du langage MQL5. En outre il y a une possibilité de créer les indicateurs personnels d'utilisateur à l'aide de la fonction iCustom(). Toutes les constantes nécessaires pour cela sont divisées en 5 groupes:

- Les constantes de prix – pour sélectionner le type de prix ou de volume, sur lesquels on calcule un indicateur;
- Les méthodes des glissantes – les méthodes insérées de lissage, utilisées dans les indicateurs;
- Les lignes des indicateurs – les identificateurs des tampons d'indicateurs à l'accès aux valeurs des indicateurs à l'aide de la fonction CopyBuffer();
- Les styles du dessin – pour indiquer un des 18 types de dessin et de détermination du style de dessin de ligne;
- Les propriétés des indicateurs d'utilisateurs – sont utilisées dans les fonctions pour le travail avec les indicateurs d'utilisateur;
- Les types des indicateurs – servent pour l'indication du type de l'indicateur technique pendant la création du handle à l'aide de la fonction IndicatorCreate();
- Les identificateurs des types de données - sont utilisés pour la détermination du type des données, transmis par le tableau comme MqlParam dans la fonction IndicatorCreate().

© 2000-2015, MetaQuotes Software Corp.

Les constantes de prix

Les indicateurs techniques exigent pour ses calculs l'indication des valeurs des prix et/ou des valeurs de volumes, sur lesquels les calculs seront exécutés. Il y a 7 identificateurs prédéterminés de l'énumération ENUM_APPLIED_PRICE, pour l'indication de la base nécessaire de prix des calculs.

ENUM_APPLIED_PRICE

Identificateur	Description
PRICE_CLOSE	Le prix de la clôture
PRICE_OPEN	Le prix de l'ouverture
PRICE_HIGH	Le prix maximum pour la période
PRICE_LOW	Le prix minimum pour la période
PRICE_MEDIAN	Le prix moyen, (high+low)/2
PRICE_TYPICAL	Le prix typique, (high+low+close)/3
PRICE_WEIGHTED	Le prix moyen pondéré, (high+low+close +close)/4

Si dans les calculs on utilise le volume, il faut indiquer une de deux valeurs de l'énumération ENUM_APPLIED_VOLUME.

ENUM_APPLIED_VOLUME

Identificateur	Description
VOLUME_TICK	Le volume de tick
VOLUME_REAL	Le volume commercial

L'indicateur technique iStochastic() deux variantes du calcul, qui peuvent utiliser:

- ou seulement les prix Close;
- ou les prix High et Low.

Pour le choix de la variante nécessaire du calcul il faut indiquer une des valeurs de l'énumération ENUM_STO_PRICE.

ENUM_STO_PRICE

Identificateur	Description
STO_LOWHIGH	La construction aux prix Low/High
STO_CLOSECLOSE	La construction aux prix Close/Close

Si l'indicateur technique pour les calculs utilise les données de prix, dont le type est détérminé par l'énumération ENUM_APPLIED_PRICE, donc le handle de n'importe quel indicateur (inséré dans le terminal ou écrit par l'utilisateur) peut être utilisée comme la série d'entrée de prix. Dans ce cas-là, les valeurs du tampon zéro de l'indicateur seront utilisées pour les calculs. Cela permet facilement de construire les valeurs d'un indicateur selon les valeurs de l'autre indicateur. Le handle de l'indicateur

© 2000-2015, MetaQuotes Software Corp.

d'utilisateur est créé par l'appel de la fonction iCustom().

Exemple:

```
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- input parameters
input int      RSIperiod=14;         // période pour le calcul RSI
input int      Smooth=8;             // période de lissage RSI
input ENUM_MA_METHOD meth=MODE_SMMA; // méthode pour le lissage
//---- plot RSI
#property indicator_label1  "RSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//---- plot RSI_Smoothed
#property indicator_label2  "RSI_Smoothed"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrNavy
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- indicator buffers
double         RSIBuffer[];          // ici nous conservons les valeurs RSI
double         RSI_SmoothedBuffer[]; // ici les valeurs RSI seront lissées
int            RSIhandle;            // descripteur sur l'indicateur RSI
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,RSIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,RSI_SmoothedBuffer,INDICATOR_DATA);
   IndicatorSetString(INDICATOR_SHORTNAME,"iRSI");
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//---
   RSIhandle=iRSI(NULL,0,RSIperiod,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[]
                )
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//--- enlevons au zéro la valeur de la dernière erreur
   ResetLastError();
//--- recevrons les données de l'indicateur RSI au tableau RSIBuffer[]
   int copied=CopyBuffer(RSIhandle,0,0,rates_total,RSIBuffer);
   if(copied<=0)
     {
      Print("On n'a pas réussi à copier les valeurs de l'indicateur RSI. Error = ",
            GetLastError(),",  copied =",copied);
      return(0);
     }
//--- créons l'indicateur moyen selon les valeurs de l'indicateur RSI
   int RSI_MA_handle=iMA(NULL,0,Smooth,0,meth,RSIhandle);
   copied=CopyBuffer(RSI_MA_handle,0,0,rates_total,RSI_SmoothedBuffer);
   if(copied<=0)
     {
      Print("On n'a pas réussi à copier l'indicateur lissé RSI. Error = ",
            GetLastError(),",  copied =",copied);
      return(0);
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Les méthodes des glissantes

Beaucoup d'indicateurs techniques sont basés sur les méthodes différentes du lissage de série des prix. Certains indicateurs standards techniques demandent l'indication du type de lissage comme un paramètre de contribution. Pour l'indication du type nécessaire de lissage, servent les identificateurs énumérés dans l'énumération ENUM_MA_METHOD.

ENUM_MA_METHOD

Identificateur	Description
MODE_SMA	La prise de moyenne simple
MODE_EMA	La prise de moyenne exponentielle
MODE_SMMA	La prise de moyenne lissée
MODE_LWMA	La prise de moyenne de ligne-pesée

Exemple:

```
double ExtJaws[];
double ExtTeeth[];
double ExtLips[];
//---- handles for moving averages
int    ExtJawsHandle;
int    ExtTeethHandle;
int    ExtLipsHandle;
//--- get MA's handles
ExtJawsHandle=iMA(NULL,0,JawsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtTeethHandle=iMA(NULL,0,TeethPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtLipsHandle=iMA(NULL,0,LipsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
```

© 2000-2015, MetaQuotes Software Corp.

Les lignes des indicateurs

Certains indicateurs techniques ont quelques tampons dessinés sur le graphique. La numération des tampons d'indicateur commence par 0. Au copiage des valeurs de l'indicateur par la fonction CopyBuffer() au tableau du type double pour certains indicateurs on peut indiquer l'indicateur de tampon copié au lieu de son numéro.

Les identificateurs des lignes des indicateurs admissibles au copiage des valeurs des indicateurs iMACD(), iRVI() et iStochastic()

Constante	Valeur	Description
MAIN_LINE	0	La ligne principale
SIGNAL_LINE	1	La ligne de signal

Les identificateurs des lignes des indicateurs admissibles au copiage des valeurs des indicateurs ADX() et ADXW()

Constante	Valeur	Description
MAIN_LINE	0	La ligne principale
PLUSDI_LINE	1	La ligne +DI
MINUSDI_LINE	2	La ligne -DI

Les identificateurs des lignes des indicateurs admissibles au copiage des valeurs des indicateurs iBands()

Constante	Valeur	Description
BASE_LINE	0	Une ligne principale
UPPER_BAND	1	Une limite supérieure
LOWER_BAND	2	Une limite inférieure

Les identificateurs des lignes des indicateurs admissibles au copiage des valeurs des indicateurs iEnvelopes() et iFractals()

Constante	Valeur	Description
UPPER_LINE	0	Une ligne supérieure
LOWER_LINE	1	Une ligne inférieure

Les identificateurs des lignes des indicateurs admissibles au copiage des valeurs des indicateurs iGator()

Constante	Valeur	Description
UPPER_HISTOGRAM	0	L'histogramme supérieur
LOWER_HISTOGRAM	2	L'histogramme inférieur

Les identificateurs des lignes des indicateurs admissibles au copiage des valeurs des indicateurs

© 2000-2015, MetaQuotes Software Corp.

iAlligator()

Constante	Valeur	Description
GATORJAW_LINE	0	La ligne de mâchoire
GATORTEETH_LINE	1	La ligne de dents
GATORLIPS_LINE	2	La ligne de lèvres

Les identificateurs des lignes des indicateurs admissibles au copiage des valeurs des indicateurs iIchimoku()

Constante	Valeur	Description
TENKANSEN_LINE	0	La ligne Tenkan-sen
KIJUNSEN_LINE	1	La ligne Kijun-sen
SENKOUSPANA_LINE	2	La ligne Senkou Span A
SENKOUSPANB_LINE	3	La ligne Senkou Span B
CHIKOUSPAN_LINE	4	La ligne Chikou Span

© 2000-2015, MetaQuotes Software Corp.

Les styles du dessin

A la création de l'indicateur d'utilisateur on peut indiquer un des 18 types de la construction graphique (le moyen d'affichage dans une fenêtre principale du graphique ou dans une sous-fenêtre du graphique), dont les valeurs sont indiquées dans l'énumération ENUM_DRAW_TYPE.

Dans un indicateur d'utilisateur il est admissible d'utiliser toutes sortes de construction/dessin des indicateurs. Chaque type de construction exige la spécification d'une à cinq tableaux globales pour le stockage des données nécessaires pour le dessin. Il est nécessaire de lier ces tableaux aux tampons d'indicateur au moyen de la fonction SetIndexBuffer(), et indiquer pour chaque tampon le type des données de l'énumération ENUM_INDEXBUFFER_TYPE.

Selon le style du dessin, on a probablement besoin d'une à quatre tampons des valeurs (marqués comme INDICATOR_DATA). Si un style admet l'alternance dynamique de couleurs (tous les styles contiennent le mot COLOR dans le nom), donc on a besoin d'un tampon de la couleur (le type INDICATOR_COLOR_INDEX est indiqué). Le tampon de couleur est toujours attachée après les valeurs correspondantes au style des tampons.

ENUM_DRAW_TYPE

Identificateur	Description	Tampons des valeurs	Tampons de la couleur
DRAW_NONE	Ne dessine pas	1	0
DRAW_LINE	La ligne	1	0
DRAW_SECTION	Les segments	1	0
DRAW_HISTOGRAM	L'histogramme de la ligne zéro	1	0
DRAW_HISTOGRAM2	L'histogramme des deux tampons d'indicateurs	2	0
DRAW_ARROW	Le dessin par les flèches	1	0
DRAW_ZIGZAG	Le style Zigzag admet les segments verticaux sur la barre	2	0
DRAW_FILLING	Le remplissage coloré entre les deux niveaux	2	0
DRAW_BARS	L'affichage comme une séquence de barres	4	0
DRAW_CANDLES	L'affichage comme des bougies	4	0
DRAW_COLOR_LINE	La ligne multicolore	1	1
DRAW_COLOR_SECTIO	Les segments	1	1

© 2000-2015, MetaQuotes Software Corp.

N	multicolores		
DRAW_COLOR_HISTOGRAM	L'histogramme multicolore de la ligne zéro	1	1
DRAW_COLOR_HISTOGRAM2	L'histogramme multicolore aux deux tampons d'indicateurs	2	1
DRAW_COLOR_ARROW	Le dessin par les flèches multicolores	1	1
DRAW_COLOR_ZIGZAG	ZigZag multicolore	2	1
DRAW_COLOR_BARS	Les barres multicolores	4	1
DRAW_COLOR_CANDLES	Les bougies multicolores	4	1

Pour la précision de l'affichage de l'aspect choisi du dessin on utilise les identificateurs énumérés dans les énumérations ENUM_PLOT_PROPERTY.

Pour les fonctions PlotIndexSetInteger() et PlotIndexGetInteger()

ENUM_PLOT_PROPERTY_INTEGER

Identificateur	Description	Type de propriété
PLOT_ARROW	Le code de la flèche pour le style DRAW_ARROW	uchar
PLOT_ARROW_SHIFT	Le décalage des flèches selon la verticale pour le style DRAW_ARROW	int
PLOT_DRAW_BEGIN	Le nombre de barres initialles sans dessin et les valeurs dans DataWindow	int
PLOT_DRAW_TYPE	Le type de la construction graphique	ENUM_DRAW_TYPE
PLOT_SHOW_DATA	Le critère de l'affichage des valeurs de la construction dans la fenêtre DataWindow	bool
PLOT_SHIFT	Le décalage de la construction graphique de l'indicateur selon l'axe du temps dans les barres	int
PLOT_LINE_STYLE	Le style de la ligne du dessin	ENUM_LINE_STYLE
PLOT_LINE_WIDTH	L'épaisseur de la ligne du dessin	int

© 2000-2015, MetaQuotes Software Corp.

PLOT_COLOR_INDEXES	Le nombre de couleurs	int
PLOT_LINE_COLOR	L'index du tampon qui contient la couleur du dessin	color le modificateur=le numéro de l'index de la couleur

Pour la fonction PlotIndexSetDouble()

ENUM_PLOT_PROPERTY_DOUBLE

Identificateur	Description	Type de propriété
PLOT_EMPTY_VALUE	Une valeur vide pour la construction, pour laquelle il n'y a aucun dessin	double

Pour la fonction PlotIndexSetString()

ENUM_PLOT_PROPERTY_STRING

Identificateur	Description	Type de propriété
PLOT_LABEL	Le nom de la série d'indicateur graphique pour l'affichage dans la fenêtre DataWindow. Pour les styles complexes graphiques demandant pour l'affichage quelques tampons d'indicateur, on peut spécifier les noms pour chaque tampon en utilisant comme délimiteur ";". L'exemple du code est dans DRAW_CANDLES	string

Les 5 styles peuvent être utilisés pour dessiner des lignes dans les indicateurs d'utilisateur. Ils sont employés seulement à l'épaisseur de la ligne 0 ou 1.

ENUM_LINE_STYLE

Identificateur	Description
STYLE_SOLID	La ligne continue
STYLE_DASH	La ligne interrompue
STYLE_DOT	La ligne pointillée
STYLE_DASHDOT	La barre- la ligne pointillée
STYLE_DASHDOTDOT	La barre- deux points

Pour mettre le style de dessin de la ligne et le type de dessin on utilise la fonction PlotIndexSetInteger(). Pour les extensions de Fibonacci il faut indiquer l'épaisseur et le style du dessin des niveaux par la fonction ObjectSetInteger().

© 2000-2015, MetaQuotes Software Corp.

Exemple:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- indicator buffers
double         MABuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- rattachement du tableau au tampon d'indicateur avec l'index 0
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
//--- spécifier le dessin de la ligne
   PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
//--- spécifier lu style pour le dessin de la ligne
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
//--- spécifier la couleur de la ligne
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);
//--- spécifier de l'épaisseur de la ligne
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
//--- spécifier la balise pour la ligne
   PlotIndexSetString(0,PLOT_LABEL,"Moving Average");
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   for(int i=prev_calculated;i<rates_total;i++)
     {
      MABuffer[i]=close[i];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Les propriétés des indicateurs d'utilisateurs

La quantité de tampons d'indicateur, lesquels on peut utiliser dans l'indicateur d'utilisateur, n'est pas limitée. Mais à chaque tableau qui est assigné à titre du tampon d'indicateur à l'aide de la fonction SetIndexBuffer(), il est nécessaire de spécifier le type de données qu'il stocke. Ce peut être une des valeurs de l'énumération ENUM_INDEXBUFFER_TYPE.

ENUM_INDEXBUFFER_TYPE

Identificateur	Description
INDICATOR_DATA	Les données pour le dessin
INDICATOR_COLOR_INDEX	Les couleurs du dessin
INDICATOR_CALCULATIONS	Les tampons auxiliaires pour les calculs intermédiaires

L'indicateur d'utilisateur a la multitude de réglages pour le confort de l'affichage et la perception. Ces réglages sont produits dans l' objectif des propriétés correspondantes de l'indicateur à l'aide des fonctions IndicatorSetDouble(), IndicatorSetInteger() et IndicatorSetString(). Les identificateurs des propriétés de l'indicateur sont énumérés dans les énumérations ENUM_CUSTOMIND_PROPERTY.

ENUM_CUSTOMIND_PROPERTY_INTEGER

Identificateur	Description	Type de propriété
INDICATOR_DIGITS	L'exactitude de l'affichage des valeurs de l'indicateur	int
INDICATOR_HEIGHT	La hauteur fixée de la fenêtre personnelle de l'indicateur (la commande du préprocesseur#property indicator_height)	int
INDICATOR_LEVELS	Le nombre de niveaux dans la fenêtre de l'indicateur	int
INDICATOR_LEVELCOLOR	La couleur de la ligne du niveau	color le modificateur =le numéro du niveau
INDICATOR_LEVELSTYLE	Le style de la ligne du niveau	ENUM_LINE_STYLE le modificateur =le numéro du niveau
INDICATOR_LEVELWIDTH	L'épaisseur de la ligne du niveau	int le modificateur =le numéro du niveau

ENUM_CUSTOMIND_PROPERTY_DOUBLE

Identificateur	Description	Type de propriété

© 2000-2015, MetaQuotes Software Corp.

INDICATOR_MINIMUM	Le minimum de la fenêtre de l'indicateur	double
INDICATOR_MAXIMUM	Le maximum de la fenêtre de l'indicateur	double
INDICATOR_LEVELVALUE	La valeur du niveau	double le modificateur=le numéro du niveau

ENUM_CUSTOMIND_PROPERTY_STRING

Identificateur	Description	Type de propriété
INDICATOR_SHORTNAME	Le nom court de l'indicateur	string
INDICATOR_LEVELTEXT	La description du niveau	string le modificateur=le numéro du niveau

Exemples:

```
//--- indicator settings
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_color2  clrRed
//--- input parameters
extern int KPeriod=5;
extern int DPeriod=3;
extern int Slowing=3;
//--- indicator buffers
double MainBuffer[];
double SignalBuffer[];
double HighesBuffer[];
double LowesBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,MainBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,HighesBuffer,INDICATOR_CALCULATIONS);
   SetIndexBuffer(3,LowesBuffer,INDICATOR_CALCULATIONS);
//--- set accuracy
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//--- set levels
```

© 2000-2015, MetaQuotes Software Corp.

```
   IndicatorSetInteger(INDICATOR_LEVELS,2);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,0,20);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,1,80);
//--- set maximum and minimum for subwindow
   IndicatorSetDouble(INDICATOR_MINIMUM,0);
   IndicatorSetDouble(INDICATOR_MAXIMUM,100);
//--- sets first bar from what index will be drawn
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,KPeriod+Slowing-2);
   PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,KPeriod+Slowing+DPeriod);
//--- set style STYLE_DOT for second line
   PlotIndexSetInteger(1,PLOT_LINE_STYLE,STYLE_DOT);
//--- name for DataWindow and indicator subwindow label
   IndicatorSetString(INDICATOR_SHORTNAME,"Stoch("+KPeriod+","+DPeriod+","+Slowing+")"
   PlotIndexSetString(0,PLOT_LABEL,"Main");
   PlotIndexSetString(1,PLOT_LABEL,"Signal");
//--- sets drawing line to empty value
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0.0);
//--- initialization done
  }
```

© 2000-2015, MetaQuotes Software Corp.

Les types des indicateurs techniques

Il y a deux moyens de créer par la programmation le handle de l'indicateur pour un ultérieur accès à ses valeurs. Le premier chemin est directement de spécifier un nom de fonction de la liste d'indicateurs techniques. Le deuxième moyen permet à l'aide de la fonction IndicatorCreate() de créer uniformément le handle de chaque indicateur par la spécification de l'identificateur de l'énumération ENUM_INDICATOR. Les deux variantes de la création du handle de l'indicateur sont égales en droits, on peut utiliser celui qui est plus confortable dans chaque cas concret à l'écriture du programme dans MQL5.

A la création de l'indicateur comme IND_CUSTOM, le champ *type* du premier élément du tableau des paramètres d'entrée MqlParam doit avoir la valeur TYPE_STRING de l'énumération ENUM_DATATYPE, et le champ *string_value* du premier élément doit contenir le nom de l'indicateur d'utilisateur.

ENUM_INDICATOR

Identificateur	Indicateur
IND_AC	Accelerator Oscillator
IND_AD	Accumulation/Distribution
IND_ADX	Average Directional Index
IND_ADXW	ADX by Welles Wilder
IND_ALLIGATOR	Alligator
IND_AMA	Adaptive Moving Average
IND_AO	Awesome Oscillator
IND_ATR	Average True Range
IND_BANDS	Bollinger Bands®
IND_BEARS	Bears Power
IND_BULLS	Bulls Power
IND_BWMFI	Market Facilitation Index
IND_CCI	Commodity Channel Index
IND_CHAIKIN	Chaikin Oscillator
IND_CUSTOM	Custom indicator
IND_DEMA	Double Exponential Moving Average
IND_DEMARKER	DeMarker
IND_ENVELOPES	Envelopes
IND_FORCE	Force Index
IND_FRACTALS	Fractals
IND_FRAMA	Fractal Adaptive Moving Average
IND_GATOR	Gator Oscillator

© 2000-2015, MetaQuotes Software Corp.

IND_ICHIMOKU	Ichimoku Kinko Hyo
IND_MA	Moving Average
IND_MACD	MACD
IND_MFI	Money Flow Index
IND_MOMENTUM	Momentum
IND_OBV	On Balance Volume
IND_OSMA	OsMA
IND_RSI	Relative Strength Index
IND_RVI	Relative Vigor Index
IND_SAR	Parabolic SAR
IND_STDDEV	Standard Deviation
IND_STOCHASTIC	Stochastic Oscillator
IND_TEMA	Triple Exponential Moving Average
IND_TRIX	Triple Exponential Moving Averages Oscillator
IND_VIDYA	Variable Index Dynamic Average
IND_VOLUMES	Volumes
IND_WPR	Williams' Percent Range

© 2000-2015, MetaQuotes Software Corp.

Les identificateurs des types de données

A la création du handle de l'indicateur par la fonction IndicatorCreate() il est nécessaire d'indiquer par le dernier paramètre le tableau du type MqlParam. Conformément, la structure MqlParam, décrivante les paramètres de l'indicateur, contient le champ spécial *type*. Ce champ contient l'information sur le type de données (le type réel, entier ou le type de ligne), qui sont transmis par l'élément concret de ce tableau. La valeur de ce champ de la structure MqlParam peut être une des valeurs de l'énumération ENUM_DATATYPE.

ENUM_DATATYPE

Identificateur	Type des données
TYPE_BOOL	bool
TYPE_CHAR	char
TYPE_UCHAR	uchar
TYPE_SHORT	short
TYPE_USHORT	ushort
TYPE_COLOR	color
TYPE_INT	int
TYPE_UINT	uint
TYPE_DATETIME	datetime
TYPE_LONG	long
TYPE_ULONG	ulong
TYPE_FLOAT	float
TYPE_DOUBLE	double
TYPE_STRING	string

Chaque élément de ce tableau décrit le paramètre correspondant d'entrée de l'indicateur technique créé, c'est pourquoi le type et l'héritage des éléments du tableau doit être strictement subi en conformité de la description.

© 2000-2015, MetaQuotes Software Corp.

L' état d'environnement

Les constantes décrivant le milieu en cours de l'exécution du programme mql5 se divisent en groupes:

- L'état du terminal de client – l'information sur le terminal de client;
- L'information sur le programme MQL5 exécuté – les propriétés du programme mql5 qui aident à contrôler son exécution;
- L'information sur l'instrument – la réception de l'information commerciale sur l'outil;
- L'information sur le compte – l'information sur le compte commercial courant;
- Statistique du test - les résultats du test de l'expert.

© 2000-2015, MetaQuotes Software Corp.

L'état du terminal de client

Les identificateurs pour la réception de l'information sur le terminal de client par les fonctions TerminalInfoInteger() et TerminalInfoString(). A titre du paramètre ces fonctions acceptent les valeurs des énumérations ENUM_TERMINAL_INFO_INTEGER et ENUM_TERMINAL_INFO_STRING conformément.

ENUM_TERMINAL_INFO_INTEGER

Identificateur	Description	Type de propriété
TERMINAL_BUILD	Le numéro de build du terminal lancé	int
TERMINAL_COMMUNITY_ACCOUNT	Флаг наличия авторизационных данных MQL5.community в терминале	bool
TERMINAL_COMMUNITY_CONNECTION	Наличие подключения к MQL5.community	bool
TERMINAL_CONNECTED	La présence de la connexion au serveur commercial	bool
TERMINAL_DLLS_ALLOWED	La permission de l'utilisation DLL	bool
TERMINAL_TRADE_ALLOWED	La permission au commerce	bool
TERMINAL_EMAIL_ENABLED	La permission de l'expédition des lettres avec l'utilisation du serveur SMTP et le nom d'utilisateur, indiqué dans les réglages du terminal	bool
TERMINAL_FTP_ENABLED	La permission de l'expédition des rapports par FTP sur le serveur indiqué pour le terminal indiqué dans les réglages du compte commercial	bool
TERMINAL_NOTIFICATIONS_ENABLED	Разрешение на отправку уведомлений на смартфон	bool
TERMINAL_MAXBARS	Le nombre maximum des barres sur le graphique	int
TERMINAL_MQID	Флаг наличия MetaQuotes ID для отправки Push-уведомлений	bool
TERMINAL_CODEPAGE	Le numéro de la page de code du langage est mis dans le terminal de client	int
TERMINAL_CPU_CORES	Le nombre de processeurs dans le système	int

© 2000-2015, MetaQuotes Software Corp.

TERMINAL_DISK_SPACE	La quantité d'espace disque libre pour le dossier MQL5 \Files du terminal (agent) en Mb	int
TERMINAL_MEMORY_PHYSICAL	La taille de la mémoire physique dans le système, en Mb	int
TERMINAL_MEMORY_TOTAL	La taille de la mémoire disponible au procès du terminal (agent), en Mb	int
TERMINAL_MEMORY_AVAILABLE	La taille de la mémoire libre du procès du terminal (agent) en Mb	int
TERMINAL_MEMORY_USED	La taille de la mémoire utilisée par le terminal (agent) en Mb	int
TERMINAL_X64	Le critère "le terminal de 64 bits"	bool
TERMINAL_OPENCL_SUPPORT	La version de l' OpenCL soutenu dans l'aspect 0x00010002 = 1.2. "0" signifit, que OpenCL n'est pas soutenu	int

ENUM_TERMINAL_INFO_DOUBLE

Identificateur	Description	Type de propriété
TERMINAL_COMMUNITY_BALANCE	Баланс пользователя в MQL5.community	double

Les opérations des fichiers peuvent être exécutées seulement dans deux répertoires; dont les chemins peuvent être obtenues en utilisant la requête des propriétés TERMINAL_DATA_PATH et TERMINAL_COMMONDATA_PATH.

ENUM_TERMINAL_INFO_STRING

Identificateur	Description	Type de propriété
TERMINAL_LANGUAGE	Langage du terminal	string
TERMINAL_COMPANY	Le nom de la compagnie	string
TERMINAL_NAME	Le nom du terminal	string
TERMINAL_PATH	Le dossier d'où le terminal a été démarré	string

© 2000-2015, MetaQuotes Software Corp.

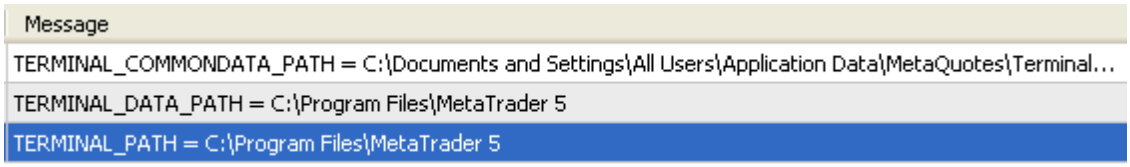
TERMINAL_DATA_PATH	Le dossier où se trouvent les données du terminal	string
TERMINAL_COMMONDATA_PATH	Le dossier commun pour tous les terminals de client, installés sur un ordinateur	string

Pour la meilleure compréhension des chemins, stockées dans les propriétés des paramètres TERMINAL_PATH, TERMINAL_DATA_PATH et TERMINAL_COMMONDATA_PATH, il est recommandé d'exécuter le script, qui informera de ces valeurs pour cette copie du terminal installé sur votre ordinateur.

Exemple: Le script rend des renseignements sur les chemins du terminal

```
//+------------------------------------------------------------------+
//|                                          Check_TerminalPaths.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   Print("TERMINAL_PATH = ",TerminalInfoString(TERMINAL_PATH));
   Print("TERMINAL_DATA_PATH = ",TerminalInfoString(TERMINAL_DATA_PATH));
   Print("TERMINAL_COMMONDATA_PATH = ",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
  }
```

A la suite de l'exécution de script les messages, ressemblants au dessin ci-dessous, seront déduits dans le Journal d'Experts.



© 2000-2015, MetaQuotes Software Corp.

L'information sur le programme MQL5 exécuté

Les constantes, pour la réception de l'information sur le programme exécuté mql5, sont énumérées à ENUM_MQL_INFO_INTEGER et ENUM_MQL_INFO_STRING.

Pour la fonction MQLInfoInteger()

ENUM_MQL_INFO_INTEGER

Identificateur	Description	Type de propriété
MQL_MEMORY_LIMIT	La quantité maximale possible de mémoire dynamique pour le programme MQL5 en Mb	int
MQL_MEMORY_USED	La taille de la mémoire utilisée par le programme MQL5 en Mb	int
MQL_PROGRAM_TYPE	Le type de programme mql5	ENUM_PROGRAM_TYPE
MQL_DLLS_ALLOWED	La permission de l'utilisation DLL pour **ce programme exécuté**	bool
MQL_TRADE_ALLOWED	La permission du commerce pour **ce programme exécuté**	bool
MQL_SIGNALS_ALLOWED	Разрешение на работу с сигналами **данной запущенной программы**	bool
MQL_DEBUG	L'indice du travail du programme exécuté au mode du débogage	bool
MQL_PROFILER	Le critère du travail du programme lancé en mode du profilage du code	bool
MQL_TESTER	L'indicatif du travail du programme exécuté au testeur	bool
MQL_OPTIMIZATION	L'indice du travail du programme exécuté en train de l'optimisation	bool
MQL_VISUAL_MODE	L'indice du travail du programme exécuté dans le mode visuel de test	bool
MQL_FRAME_MODE	Le critère du travail de l'expert lancé sur le graphique enmode du rassemblement des trames des résultats de l'optimisation	bool
MQL_LICENSE_TYPE	Le type de la licence du module EX5. La licence se	ENUM_LICENSE_TYPE

© 2000-2015, MetaQuotes Software Corp.

	rapporte notamment à ce module EX5, d'où on fait la demande à l'aide de MQLInfoInteger(MQL_LICENSE_TYPE).	

Pour la fonction MQLInfoString()

ENUM_MQL_INFO_STRING

Identificateur	Description	Type de propriété
MQL_PROGRAM_NAME	Le nom du programme mql5 exécuté	string
MQL_PROGRAM_PATH	Le chemin pour **ce programme exécuté**	string

Les valeurs d'énumération ENUM_PROGRAM_TYPE se sont destinées pour recevoir les renseignements sur le type du programme exécuté.

ENUM_PROGRAM_TYPE

Identificateur	Description
PROGRAM_SCRIPT	Le script
PROGRAM_EXPERT	L'expert
PROGRAM_INDICATOR	L'indicateur

ENUM_LICENSE_TYPE

Identificateur	La description
LICENSE_FREE	La version gratuite non limitée
LICENSE_DEMO	La version démo du produit payant du Market Fonctionne seulement dans le testeur des stratégies
LICENSE_FULL	Купленная лицензионная версия допускает не менее 5 активаций. Продавец может увеличить разрешенное число активаций
LICENSE_TIME	Версия с ограниченной по времени лицензией

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
ENUM_PROGRAM_TYPE mql_program=(ENUM_PROGRAM_TYPE)MQLInfoInteger(MQL_PROGRAM_TYPE);
switch(mql_program)
  {
   case PROGRAM_SCRIPT:
     {
      Print(__FILE__+" is script");
      break;
     }
   case PROGRAM_EXPERT:
     {
      Print(__FILE__+" is Expert Advisor");
      break;
     }
   case PROGRAM_INDICATOR:
     {
      Print(__FILE__+" is custom indicator");
      break;
     }
   default:Print("MQL5 program type value is ",mql_program);
  }
```

© 2000-2015, MetaQuotes Software Corp.

L'information sur l'instrument

Pour la réception de l'information courante de marché servent les fonctions SymbolInfoInteger(), SymbolInfoDouble() et SymbolInfoString(). A titre du deuxième paramètre de ces fonctions il est admissible de transmettre un des identificateurs des énumérations ENUM_SYMBOL_INFO_INTEGER, ENUM_SYMBOL_INFO_DOUBLE et ENUM_SYMBOL_INFO_STRING conformément.

Pour la fonction SymbolInfoInteger()

ENUM_SYMBOL_INFO_INTEGER

Identificateur	Description	Type de propriété
SYMBOL_SELECT	Le symbole est choisi dans Market Watch	bool
SYMBOL_SESSION_DEALS	Le nombre de marchés dans la session courante	long
SYMBOL_SESSION_BUY_ORDERS	Le nombre total d'ordres à l'achat pour le moment	long
SYMBOL_SESSION_SELL_ORDERS	Le nombre total d'ordres à la vente pour le moment	long
SYMBOL_VOLUME	Le volume - le volume dans le dernier marché	long
SYMBOL_VOLUMEHIGH	Le volume maximum par jour	long
SYMBOL_VOLUMELOW	Le volume minimum par jour	long
SYMBOL_TIME	Le temps de la dernière cotation	datetime
SYMBOL_DIGITS	Le nombre de signes après la virgule	int
SYMBOL_SPREAD	La valeur de spread dans les points	int
SYMBOL_SPREAD_FLOAT	L'indication du spread flottant	bool
SYMBOL_TICKS_BOOKDEPTH	Le nombre maximal des demandes montrées au profondeur de marché. Pour les instruments qui n'ont pas de tour des demandes, la valeur est égal au 0	int
SYMBOL_TRADE_CALC_MODE	La mode de calcul des prix de contrat	ENUM_SYMBOL_CALC_MODE
SYMBOL_TRADE_MODE	Le type d'exécution d'ordre	ENUM_SYMBOL_TRADE_MODE
SYMBOL_START_TIME	La date du commencement des enchères selon l'instrument (d'habitude il est utilisé pour	datetime

© 2000-2015, MetaQuotes Software Corp.

	les contrats à terme)	
SYMBOL_EXPIRATION_TIME	La date de l'achèvement des enchères selon l'instrument (d'habitude il est utilisé pour les contrats à terme)	datetime
SYMBOL_TRADE_STOPS_LEVEL	L'alinéa minimal dans les points du prix courant de la clôture pour l'installation des ordres Stop	int
SYMBOL_TRADE_FREEZE_LEVEL	La distance pour bloquer des opérations commerciales (dans les points)	int
SYMBOL_TRADE_EXEMODE	La mode d'exécution d'affaire	ENUM_SYMBOL_TRADE_EXECUTION
SYMBOL_SWAP_MODE	Le modèle de calcul de swap	ENUM_SYMBOL_SWAP_MODE
SYMBOL_SWAP_ROLLOVER3DAYS	Le jour de la semaine pour le calcul de swap triple	ENUM_DAY_OF_WEEK
SYMBOL_EXPIRATION_MODE	Les drapeaux des modes permis de l'expiration de l'ordre	int
SYMBOL_FILLING_MODE	Les drapeaux des modes permis du remplissage de l'ordre	int
SYMBOL_ORDER_MODE	Les drapeaux des types permis de l'ordre	int
SYMBOL_OPTION_MODE	Тип опциона	ENUM_SYMBOL_OPTION_MODE
SYMBOL_OPTION_RIGHT	Право опциона (Call/Put)	ENUM_SYMBOL_OPTION_RIGHT

Pour la fonction SymbolInfoDouble()

ENUM_SYMBOL_INFO_DOUBLE

Identificateur	Description	Type de propriété
SYMBOL_BID	Bid - une meilleure proposition pour la vente	double
SYMBOL_BIDHIGH	Bid maximum par jour	double
SYMBOL_BIDLOW	Bid minimum par jour	double
SYMBOL_ASK	Ask - une meilleure proposition pour l'achat	double
SYMBOL_ASKHIGH	Ask maximum par jour	double
SYMBOL_ASKLOW	Ask minimum par jour	double

© 2000-2015, MetaQuotes Software Corp.

SYMBOL_LAST	Prix de la dernière affaire	double
SYMBOL_LASTHIGH	Last maximum par jour	double
SYMBOL_LASTLOW	Last minimum par jour	double
SYMBOL_OPTION_STRIKE	Цена исполнения опциона. Это цена, по которой покупатель опциона может купить (при опционе Call) или продать (при опционе Put) базовый актив, а продавец опциона соответственно обязан продать или купить соответствующее количество базового актива.	double
SYMBOL_POINT	La valeur de point de symbole	double
SYMBOL_TRADE_TICK_VALUE	La valeur SYMBOL_TRADE_TICK_VALUE_PROFIT	double
SYMBOL_TRADE_TICK_VALUE_PROFIT	Le coût calculé du tick pour la position profitable	double
SYMBOL_TRADE_TICK_VALUE_LOSS	Le coût calculé du tick pour la position déficitaire	double
SYMBOL_TRADE_TICK_SIZE	Le changement minimal du prix	double
SYMBOL_TRADE_CONTRACT_SIZE	Le montant du contrat commercial	double
SYMBOL_VOLUME_MIN	Le volume minimal pour une affaire	double
SYMBOL_VOLUME_MAX	Le volume maximal pour une affaire	double
SYMBOL_VOLUME_STEP	Un pas minimal du changement du volume pour une affaire	double
SYMBOL_VOLUME_LIMIT	Un volume cumulé au maximum admissible pour le symbole donné de la position ouverte et des ordres remis dans une direction (l'achat ou la vente). Par exemple, à la limitation aux 5 lots on peut avoir la position ouverte sur l'achat par le volume de 5 lots et exposer l'ordre remis Sell Limit par le volume de 5 lots. Mais on ne peut pas exposer	double

© 2000-2015, MetaQuotes Software Corp.

	l'ordre remis Buy Limit (puisque un volume cumulé dans une direction excédera la limitation) ou exposer Sell Limit par le volume plus de 5 lots.	
SYMBOL_SWAP_LONG	La valeur de swap à l'achat	double
SYMBOL_SWAP_SHORT	La valeur de swap à la vente	double
SYMBOL_MARGIN_INITIAL	La marge initiale (initiante) désigne le montant des moyens nécessaires hypothécaires en devise de marge pour l'ouverture de la position par le volume qui correspond à un lot. Elle est utilisée au contrôle des moyens du client à l'entrée au marché.	double
SYMBOL_MARGIN_MAINTENANCE	Le marge de maintenance selon l'instrument. En cas si elle est spécifiée – on indique le montant de la marge en devise de marge de l'instrument qui est retenue d'un lot. Elle est utilisé au contrôle des moyens du client au changement de l'état du compte du client. Si la marge de maintenance est égale au 0, on utilise la marge initiale.	double
SYMBOL_SESSION_VOLUME	Le volume total des marchés à la session courante	double
SYMBOL_SESSION_TURNOVER	Le chiffre d'affaires total des marchés à la session courante	double
SYMBOL_SESSION_INTEREST	Le volume total des positions ouvertes	double
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Le volume total des ordres à l'achat pour le moment	double
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Le volume total des ordres à la vente pour le moment	double
SYMBOL_SESSION_OPEN	Le prix de l'ouverture de la session	double
SYMBOL_SESSION_CLOSE	Le prix de la clôture de la session	double

© 2000-2015, MetaQuotes Software Corp.

SYMBOL_SESSION_AW	Le prix moyen pondéré de la session	double
SYMBOL_SESSION_PRICE_SETTLEMENT	Le prix de la livraison à la session courante	double
SYMBOL_SESSION_PRICE_LIMIT_MIN	La valeur au minimum admissible du prix de la session	double
SYMBOL_SESSION_PRICE_LIMIT_MAX	La valeur au maximum admissible du prix de la session	double

Pour la fonction SymbolInfoString()

ENUM_SYMBOL_INFO_STRING

Identificateur	Description	Type de propriété
SYMBOL_BASIS	Имя базового актива для производного инструмента	string
SYMBOL_CURRENCY_BASE	La devise de base de l'instrument	string
SYMBOL_CURRENCY_PROFIT	La devise du bénéfice	string
SYMBOL_CURRENCY_MARGIN	La devise des moyens hypothécaires	string
SYMBOL_BANK	La source de la cotation en cours	string
SYMBOL_DESCRIPTION	La description de ligne du caractère	string
SYMBOL_ISIN	Le nom du symbole commercial dans le système des codes internationaux identificatoires des valeurs mobilières— ISIN (International Securities Identification Number). Le numéro international d'identification de la valeur mobilière - c'est un code alphanumérique à 12 chiffres qui identifie la valeur mobilière absolument. La présence de la propriété donnée du symbole est définie au côté du serveur commercial.	string
SYMBOL_PATH	Le chemin dans l'arbre des caractères	string

© 2000-2015, MetaQuotes Software Corp.

Pour chaque instrument financier on peut indiquer quelques modes de la durée (expiration) des ordres remis. À chaque mode on compare le drapeau, les drapeaux peuvent être combinés par l'opération logique **OU** (|), par exemple, SYMBOL_EXPIRATION_GTC|SYMBOL_EXPIRATION_SPECIFIED. Pour vérifier le permis de mode particulier pour l'instrument, il faut comparer le résultat de **ET** (&) logique avec le drapeau de la mode.

Si on indique le drapeau SYMBOL_EXPIRATION_SPECIFIED pour le caractère, à l'expédition de l'ordre remis on peut concrètement indiquer jusqu'à quel moment l'ordre donné remis fonctionne.

Identificateur	Valeur	Description
SYMBOL_EXPIRATION_GTC	1	L'ordre est valable sans limitation par le temps jusqu'à son annulation évidente
SYMBOL_EXPIRATION_DAY	2	L'ordre est valable jusqu'à la fin du jour
SYMBOL_EXPIRATION_SPECIFIED	4	Le délai de l'expiration est indiqué dans l'ordre
SYMBOL_EXPIRATION_SPECIFIED_DAY	8	Le jour de l'expiration est indiqué dans l'ordre

Exemple:

```
//+------------------------------------------------------------------+
//| vérifie le permis de la mode indiquée de l'expiration            |
//+------------------------------------------------------------------+
bool IsExpirationTypeAllowed(string symbol,int exp_type)
  {
//--- recevrons la valeur de propriété décrivant les modes admissibles de l'expiratior
   int expiration=(int)SymbolInfoInteger(symbol,SYMBOL_EXPIRATION_MODE);
//--- rendons true, si le mode exp_type est permis
   return((expiration & exp_type)==exp_type);
  }
```

A l'expédition de l'ordre on peut indiquer la politique du remplissage du volume déclaré dans l'ordre commercial. Les variantes admissibles de l'exécution de l'ordre par le volume pour chaque caractère sont indiquées au tableau. Pour chaque instrument on peut installer plus d'une mode, mais quelques modes dans la combinaison des drapeaux. La combinaison de drapeaux est exprimée par l'opération **OU** (|) logique, par exemple, SYMBOL_FILLING_FOK|SYMBOL_FILLING_IOC. Pour vérifier le permis de mode particulier pour l'instrument, il faut comparer le résultat de **ET** (&) logique avec le drapeau de la mode.

Remplissage	Identificateur	Valeur	Description
Tout/Rien	SYMBOL_FILLING_FOK	1	Cette politique de l'exécution signifie que l'ordre peut être

© 2000-2015, MetaQuotes Software Corp.

			exceptionnellement exécuté dans le volume indiqué. Si au marché au moment donné il n'y a pas de volume suffisant de l'instrument financier, l'ordre ne sera pas exécuté. Le volume nécessaire peut être fait de quelques propositions accessibles au moment donné sur le marché.
Tout /Partiellement	SYMBOL_FILLING_IOC	2	Dans ce cas le trader accepte de passer le marché par le volume au maximum accessible sur le marché dans la limite du volume indiqué dans l'ordre. En cas d'impossibilité de l'exécution complète l'ordre sera exécuté sur le volume accessible, et le volume non exécuté de l'ordre sera annulé. La possibilité de l'utilisation des ordres IOC est définie sur le serveur commercial.
Rendre	L'identificateur manque		Ce mode est utilisé pour les ordres de marché (Buy et Sell), limites et de stop de limite et seulement en modes "Exécution selon le marché" et "Exécution de bourse". En cas de l'exécution partielle l'ordre de marché ou limité avec le volume résiduel n'est pas remis et continue de fonctionner.

© 2000-2015, MetaQuotes Software Corp.

En modes d'exécution "Sur la demande" et "Immédiat" pour les ordres de marché on utilise toujours la politique du remplissage Tout/Rien, et pour les ordres limites - le mode "Rendre". Dans ce cas, à l'envoi des ordres par les fonctionsOrderSend ou OrderSendAsync on peut ne pas indiquer le type du remplissage pour eux.

En modes de l'exécution "Selon le marché" et "Boursier" la politique du remplissage "Rendre" est toujours permise pour tous les types des ordres. La résolution des autres types est vérifiée à l'aide des propriétés SYMBOL_FILLING_FOK et SYMBOL_FILLING_IOC.

Exemple:

```
//+------------------------------------------------------------------+
//| vérifie le permis de la mode indiquée de l'expiration            |
//+------------------------------------------------------------------+
bool IsFillingTypeAllowed(string symbol,int fill_type)
  {
//--- recevrons la valeur de la propriété décrivant le mode du remplissage
   int filling=(int)SymbolInfoInteger(symbol,SYMBOL_FILLING_MODE);
//--- rendons true, si le mode fill_type est permis
   return((filling & fill_type)==fill_type);
  }
```

A l'expéditionde la demande commerciale par la fonction OrderSend() pour certaines opérations il est nécessaire d'indiquer le type de l'ordre de l'énumération ENUM_ORDER_TYPE. Pas tous les types d'ordres peuvent être permis pour un instrument financier concret, la propriétéSYMBOL_ORDER_MODE décrit les drapeaux des types permis des ordres.

Identificateur	La valeur	La description
SYMBOL_ORDER_MARKET	1	Les ordres de marché sont permis (Buy et Sell)
SYMBOL_ORDER_LIMIT	2	Les ordres limites sont permis (Buy Limit et Sell Limit)
SYMBOL_ORDER_STOP	4	Les ordres stop sont permis (Buy Stop et Sell Stop)
SYMBOL_ORDER_STOP_LIMIT	8	Les ordres stop limites sont permis (Buy Stop Limit et Sell Stop Limit)
SYMBOL_ORDER_SL	16	L'établissement de Stop Loss est permis
SYMBOL_ORDER_TP	32	L'établissement de Take Profit est permis

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| La fonction déduit les types des ordres permis pour le symbole sur l'impression
//+------------------------------------------------------------------+
void Check_SYMBOL_ORDER_MODE(string symbol)
  {
//--- recevrons la valeur de la propriété décrivant les types permis des ordres
   int symbol_order_mode=(int)SymbolInfoInteger(symbol,SYMBOL_ORDER_MODE);
//--- la vérification aux ordres de marché (Market Execution)
   if((SYMBOL_ORDER_MARKET&symbol_order_mode)==SYMBOL_ORDER_MARKET)
      Print(symbol+": Les ordres de marché sont permis (Buy et Sell)");
//--- la vérification aux ordres du type Limit
   if((SYMBOL_ORDER_LIMIT&symbol_order_mode)==SYMBOL_ORDER_LIMIT)
      Print(symbol+": Les ordres Buy Limit et Sell Limit sont permis");
//--- la vérification aux ordres du type Stop
   if((SYMBOL_ORDER_STOP&symbol_order_mode)==SYMBOL_ORDER_STOP)
      Print(symbol+": Les ordres Buy Stop et Sell Stop sont permis");
//--- la vérification aux ordres du type Stop Limit
   if((SYMBOL_ORDER_STOP_LIMIT&symbol_order_mode)==SYMBOL_ORDER_STOP_LIMIT)
      Print(symbol+": Les ordres Buy Stop Limit et Sell Stop Limit sont permis");
//--- la vérification de la possibilité de l'établissement des ordres Stop Loss
   if((SYMBOL_ORDER_SL&symbol_order_mode)==SYMBOL_ORDER_SL)
      Print(symbol+": Les ordres Stop Loss sont permis");
//--- la vérification de la possibilité de l'établissement des ordres Take Profit
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_TP)
      Print(symbol+": Les ordres Take Profit sont permis");
//---
  }
```

Pour la réception de l'information sur le moyen du calcul de la valeur des moyens hypothécaires selon l'instrument (du montant des exigences de marge) est destiné l'énumération ENUM_SYMBOL_CALC_MODE.

ENUM_SYMBOL_CALC_MODE

Identificateur	Description	Formule
SYMBOL_CALC_MODE_FOREX	Forex mode - le calcul du profit et la marge pour Forex	Margin: Lots*Contract_Size/ Leverage Profit: (close_price-open_price) *Contract_Size*Lots
SYMBOL_CALC_MODE_FUTURES	Futures mode - le calcul du gage et du profit pour les futures	Margin: Lots *InitialMargin*Percentage/100 Profit: (close_price-open_price)*TickPrice/ TickSize*Lots
SYMBOL_CALC_MODE_CFD	CFD mode - le calcul du gage et du profit pour CFD	Margin: Lots *ContractSize*MarketPrice*Percentage/100 Profit: (close_price-open_price) *Contract_Size*Lots

© 2000-2015, MetaQuotes Software Corp.

SYMBOL_CALC_MODE_CFDINDEX	CFD index mode - le calcul du gage et du profit pour CFD par les index	Margin: (Lots*ContractSize*MarketPrice)*TickPrice/TickSize Profit: (close_price-open_price) *Contract_Size*Lots
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD Leverage mode - le calcul du gage et du profit pour CFD au commerce d'effet de levier	Margin: (Lots*ContractSize*MarketPrice*Percentage)/Leverage Profit: (close_price-open_price) *Contract_Size*Lots
SYMBOL_CALC_MODE_EXCH_STOCKS	Exchange mode - le calcul de la garantie et des bénéfices pour la commerce des papiers monétaires sur la bourse	Margin: Lots*ContractSize*OpenPrice Profit: (close_price-open_price) *Contract_Size*Lots
SYMBOL_CALC_MODE_EXCH_FUTURES	Futures mode - le calcul de la garantie et des bénéfices pour la commerce des contrats à terme sur la bourse	Margin: Lots*InitialMargin ou Lots*MaintenanceMargin Profit: (close_price-open_price)*Lots*TickPrice/TickSize
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS Futures mode - le calcul de la garantie et des bénéfices pour la commerce des contrats à terme sur FORTS. Le montant de la marge peut diminuer de la valeur de l'écart MarginDiscount selon les règles suivantes: 1. Si le prix de la position longue (les ordres sur l'achat) est moins que le prix calculé, alors MarginDiscount = Lots*((PriceSettle-PriceOrder) *TickPrice/TickSize) 2. Si le prix de la position courte (les ordres sur l'achat) est plus que le prix calculé, alors MarginDiscount = Lots*((PriceOrder-PriceSettle) *TickPrice/TickSize) où: ○ PriceSettle - le prix calculé (de clearing) de la session précédente; ○ PriceOrder - le prix moyen pondéré de la position ou le prix de l'ouverture indiqué dans l'ordre (la demande);	Margin: Lots*InitialMargin ou Lots*MaintenanceMargin Profit: (close_price-open_price)*Lots*TickPrice/TickSize

© 2000-2015, MetaQuotes Software Corp.

	○ TickPrice - le prix du tick (le coût du changement du prix d'un point) ○ TickSize - la taille du tick (le pas minimal du changement du prix)	
SYMBOL_CALC_MODE_SERV_COLLATERAL	Collateral mode - a symbol is used as a non-tradable asset on a trading account. The market value of an open position is calculated based on the volume, current market price, contract size and liquidity ratio. The value is included into Assets, which are added to Equity. Open positions of such symbols increase the Free Margin amount and are used as additional margin (collateral) for open positions of tradable instruments.	Margin: no Profit: no Market Value: Lots*ContractSize*MarketPrice*LiqudityRate

Il y a quelques modes du commerce selon les instruments financiers. L'information sur les modes du commerce sur l'instrument concret est affichée dans les valeurs de l'énumération ENUM_SYMBOL_TRADE_MODE.

ENUM_SYMBOL_TRADE_MODE

Identificateur	Description
SYMBOL_TRADE_MODE_DISABLED	Le commerce selon le symbole est interdit
SYMBOL_TRADE_MODE_LONGONLY	On permet seulement les achats
SYMBOL_TRADE_MODE_SHORTONLY	On permet seulement les ventes
SYMBOL_TRADE_MODE_CLOSEONLY	On permet seulement les opérations de la clôture des positions
SYMBOL_TRADE_MODE_FULL	Il n'y a pas de limitations sur les transactions commerciales

Les modes possibles de conclusion de transactions sur l'instrument particulier sont déterminés dans l'énumération ENUM_SYMBOL_TRADE_EXECUTION.

ENUM_SYMBOL_TRADE_EXECUTION

Identificateur	Description
SYMBOL_TRADE_EXECUTION_REQUEST	Le commerce sur la demande

© 2000-2015, MetaQuotes Software Corp.

SYMBOL_TRADE_EXECUTION_INSTANT	Le commerce sur prix de courant
SYMBOL_TRADE_EXECUTION_MARKET	L'exécution des ordres selon le marché
SYMBOL_TRADE_EXECUTION_EXCHANGE	L'exécution boursière

Les moyens du calcul des swaps au transfert de la position sont indiqués dans l'énumération ENUM_SYMBOL_SWAP_MODE. Le moyen du calcul des swaps définit les unités de mesure des paramètresSYMBOL_SWAP_LONG et SYMBOL_SWAP_SHORT. Par exemple, si les swaps sont calculés en devise du dépôt du client, dans les paramètres le volume des swaps calculés est indiqué notamment en devise du dépôt du client.

ENUM_SYMBOL_SWAP_MODE

Identificateur	Description
SYMBOL_SWAP_MODE_DISABLED	Pas de swap
SYMBOL_SWAP_MODE_POINTS	Les swaps sont calculés dans les points
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Les swaps sont calculés dans l'argent en devise de base du symbole
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Les swaps sont calculés dans l'argent en devise de marge du symbole
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Les swaps sont calculés dans l'argent en devise du dépôt du client
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Les swaps sont calculés dans les pour-cent annuels du prix de l'instrument au moment du calcul du swap (le mode bancaire – 360 jours en année)
SYMBOL_SWAP_MODE_INTEREST_OPEN	Les swaps sont calculés dans les pour-cent annuels du prix de la position de l'ouverture selon le symbole (le mode bancaire – 360 jours en année)
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Les swaps sont calculés par l'ouverture répétée de la position. A la fin du jour commercial la position se ferme forcément. Le lendemain la position de l'ouverture répétée selon le prix de la clôture + / - le nombre indiqué de points (dans les paramètres SYMBOL_SWAP_LONG et SYMBOL_SWAP_SHORT)
SYMBOL_SWAP_MODE_REOPEN_BID	Les swaps sont calculés par l'ouverture répétée de la position. A la fin du jour commercial la position se ferme forcément. Le lendemain la position s'ouvre de nouveau selon le prix courant Bid +/- le nombre indiqué de points (dans les paramètres SYMBOL_SWAP_LONG et SYMBOL_SWAP_SHORT)

© 2000-2015, MetaQuotes Software Corp.

Pour l'indication du jour de la semaine se sont destinées les valeurs de l'énumération ENUM_DAY_OF_WEEK.

ENUM_DAY_OF_WEEK

Identificateur	Description
SUNDAY	Dimanche
MONDAY	Lundi
TUESDAY	Mardi
WEDNESDAY	Mercredi
THURSDAY	Jeudi
FRIDAY	Vendredi
SATURDAY	Samedi

Опцион - это контракт, который дает право, но не обязанность купить или продать базовый актив (товар, акцию, фьючерс и т.д.) по фиксированной цене в течении жизни опциона или в определенный момент времени. Для описания свойств опционов предназначены перечисления, которые описывают тип опциона и право, который он представляет.

ENUM_SYMBOL_OPTION_RIGHT

Идентификатор	Описание
SYMBOL_OPTION_RIGHT_CALL	Опцион, дающий право купить актив по фиксированной цене
SYMBOL_OPTION_RIGHT_PUT	Опцион, дающий право продать актив по фиксированной цене

ENUM_SYMBOL_OPTION_MODE

Идентификатор	Описание
SYMBOL_OPTION_MODE_AMERICAN	Европейский тип опциона - может быть погашен только в указанную дату (дата истечения срока, дата исполнения, дата погашения)
SYMBOL_OPTION_MODE_EUROPEAN	Американский тип опциона - может быть погашен в любой день до истечения срока опциона. Для такого типа задаётся период, в течение которого покупатель может исполнить данный опцион

© 2000-2015, MetaQuotes Software Corp.

L'information sur le compte

Pour obtenir des renseignements sur le compte courant il y a plusieurs fonctions AccountInfoInteger(), AccountInfoDouble() et AccountInfoString(). A titre du paramètre ces fonctions acceptent les valeurs des énumérations correspondantes ENUM_ACCOUNT_INFO.

Pour la fonction AccountInfoInteger()

ENUM_ACCOUNT_INFO_INTEGER

Identificateur	Description	Type de propriété
ACCOUNT_LOGIN	Le numéro de compte	long
ACCOUNT_TRADE_MODE	Account trade mode	ENUM_ACCOUNT_TRADE_MODE
ACCOUNT_LEVERAGE	Le montant de l'épaule accordée	long
ACCOUNT_LIMIT_ORDERS	La quantité au maximum admissible d'ordres actifs remis	int
ACCOUNT_MARGIN_SO_MODE	Le mode de la spécification du niveau minimum admissible des moyens hypothécaires	ENUM_ACCOUNT_STOPOUT_MODE
ACCOUNT_TRADE_ALLOWED	Le commerce permis pour le compte courant	bool
ACCOUNT_TRADE_EXPERT	Le commerce permis pour l'expert	bool

Pour la fonction AccountInfoDouble()

ENUM_ACCOUNT_INFO_DOUBLE

Identificateur	Description	Type de propriété
ACCOUNT_BALANCE	La balance du compte en devise du dépôt	double
ACCOUNT_CREDIT	Le montant du crédit accordé en devise du dépôt	double
ACCOUNT_PROFIT	Le montant du bénéfice courant sur le compte en devise du dépôt	double
ACCOUNT_EQUITY	La valeur des moyens propres sur le compte en devise du dépôt	double
ACCOUNT_MARGIN	Le montant des moyens réservés hypothécaires sur le compte en devise du dépôt	double

© 2000-2015, MetaQuotes Software Corp.

ACCOUNT_MARGIN_FREE	Le montant des moyens libres sur le compte en devise du dépôt, accessible pour l'ouverture de la position	double
ACCOUNT_MARGIN_LEVEL	Le niveau des moyens hypothécaires sur le compte en pourcentage	double
ACCOUNT_MARGIN_SO_CALL	Margin call level. Depending on the set ACCOUNT_MARGIN_SO_MODE is expressed in percents or in the deposit currency. ACCOUNT_MARGIN_SO_MODE s'exprime en pourcentage ou en devise du dépôt	double
ACCOUNT_MARGIN_SO_SO	Margin stop out level. Depending on the set ACCOUNT_MARGIN_SO_MODE is expressed in percents or in the deposit currency. ACCOUNT_MARGIN_SO_MODE s'exprime en pourcentage ou en devise du dépôt	double
ACCOUNT_MARGIN_INITIAL	Размер средств, зарезервированных на счёте, для обеспечения гарантийной суммы по всем отложенным ордерам	double
ACCOUNT_MARGIN_MAINTENANCE	Размер средств, зарезервированных на счёте, для обеспечения минимальной суммы по всем открытым позициям	double
ACCOUNT_ASSETS	Текущий размер активов на счёте	double
ACCOUNT_LIABILITIES	Текущий размер обязательств на счёте	double
ACCOUNT_COMMISSION_BLOCKED	Текущая сумма заблокированных комиссий по счёту	double

Pour la fonction AccountInfoString()

ENUM_ACCOUNT_INFO_STRING

Identificateur	Description	Type de propriété

© 2000-2015, MetaQuotes Software Corp.

ACCOUNT_NAME	Le nom du client	string
ACCOUNT_SERVER	Le nom du serveur commercial	string
ACCOUNT_CURRENCY	La devise du dépôt	string
ACCOUNT_COMPANY	Le nom d'une compagnie qui sert le compte	string

Il y a quelques types des comptes, qui peuvent être ouverts sur le serveur commercial. Pour savoir le type du compte, sur lequel le programme MQL5 travaille, l'énumération ENUM_ACCOUNT_TRADE_MODE est destiné.

ENUM_ACCOUNT_TRADE_MODE

Identificateur	Description
ACCOUNT_TRADE_MODE_DEMO	Le compte commercial de démonstration
ACCOUNT_TRADE_MODE_CONTEST	Le compte de concours commercial
ACCOUNT_TRADE_MODE_REAL	Le compte réel commercial

La situation de la clôture forcée Stop Out apparaît pour maintenir des positions ouvertes au manque des moyens personnels. Le niveau minimum de la marge, quand arrive Stop Out, peut être spécifié en pourcentage ou en valeur exprimée en argent. Onpeut savoir quel mode est spécifié pour ce compte à l'aide de l'énumération ENUM_ACCOUNT_STOPOUT_MODE.

ENUM_ACCOUNT_STOPOUT_MODE

Identificateur	Description
ACCOUNT_STOPOUT_MODE_PERCENT	Le niveau est donné en pourcentage
ACCOUNT_STOPOUT_MODE_MONEY	Le niveau est donné dans l'argent

L'exemple du script, déduisant l'information brève sur le compte.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- le nom de la compagnie
   string company=AccountInfoString(ACCOUNT_COMPANY);
//--- le nom du client
   string name=AccountInfoString(ACCOUNT_NAME);
//--- le numéro du compte
   long login=AccountInfoInteger(ACCOUNT_LOGIN);
//--- le nom du serveur
   string server=AccountInfoString(ACCOUNT_SERVER);
//---la devise du compte
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//---un compte démo, de concours ou réel
```

© 2000-2015, MetaQuotes Software Corp.

```
   ENUM_ACCOUNT_TRADE_MODE account_type=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(AC
//--- maintenant transformons la valeur de l'énumération en aspect clair
   string trade_mode;
   switch(account_type)
     {
      case  ACCOUNT_TRADE_MODE_DEMO:
         trade_mode="demo";
         break;
      case  ACCOUNT_TRADE_MODE_CONTEST:
         trade_mode="de concours";
         break;
      default:
         trade_mode="réel";
         break;
     }
//--- le niveau Stop Out est spécifié en pourcentage ou en valeur exprimée en argent
   ENUM_ACCOUNT_STOPOUT_MODE stop_out_mode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoInteg
//--- recevrons les valeurs des niveaux au cours de lesquels arrive Margin Call et Sto
   double margin_call=AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL);
   double stop_out=AccountInfoDouble(ACCOUNT_MARGIN_SO_SO);
//--- déduisons l'information brève sur le compte
   PrintFormat("Le compte du client '%s' #%d %s est ouvert dans '%s' sur le serveur '%
               name,login,trade_mode,company,server);
   PrintFormat("La devise du compte - %s, le niveau MarginCall et StopOut est spécifié
               currency,(stop_out_mode==ACCOUNT_STOPOUT_MODE_PERCENT)?"en pourcentage"
   PrintFormat("Le niveau MarginCall=%G, le niveau StopOut=%G",margin_call,stop_out);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Statistique du test

Après la fin du test on calcule les paramètres statistiques des résultats du commerce selon la multitude de paramètres. On peut recevoir les valeurs des paramètres à l'aide de la fonctionTesterStatistics(), ayant indiqué l'identificateur du paramètre de l'énumération ENUM_STATISTICS.

Bien que les paramètres de deux types - int et double - sont utilisés pour calculer la statistique, la fonction rend toutes les valeurs dans l'aspect double. Toutes les valeurs statistiques qui ont le type double s'expriment endevise du dépôt par défaut, si l'autre n'est pas spécifié.

ENUM_STATISTICS

Identificateur	Description du paramètre statistique	Type
STAT_INITIAL_DEPOSIT	La valeur du dépôt initial	double
STAT_WITHDRAWAL	Le nombre de moyens déduits du compte	double
STAT_PROFIT	Le bénéfice net après le test, la somme STAT_GROSS_PROFIT et STAT_GROSS_LOSS (STAT_GROSS_LOSS est toujours moins ou égal au zéro)	double
STAT_GROSS_PROFIT	Le bénéfice total, la somme de tous les trades profitables (positifs). La valeur est plus ou égal au zéro	double
STAT_GROSS_LOSS	Une perte totale, la somme de tous les trades déficitaires (négatifs). La valeur est moins ou égal au zéro	double
STAT_MAX_PROFITTRADE	Un bénéfice maximal – la plus grande valeur parmi tous les trades profitables. La valeur est plus ou égal au zéro	double
STAT_MAX_LOSSTRADE	Une perte maximale - la valeur la plus petite parmi tous les trades déficitaires. La valeur est moins ou égal au zéro	double
STAT_CONPROFITMAX	Un bénéfice maximal dans la plus grande valeur parmi toutes les trades profitables. La valeur est plus ou égal au zéro	double
STAT_CONPROFITMAX_TRADES	Le nombre de trades qui ont	int

© 2000-2015, MetaQuotes Software Corp.

	forméSTAT_CONPROFITMAX (un bénéfice maximal au héritage des trades profitables)	
STAT_MAX_CONWINS	Le bénéfice total dans la plus longue série des trades profitables	double
STAT_MAX_CONPROFIT_TRADES	Le nombre de trades dans la plus longue série des trades profitablesSTAT_MAX_CONWINS	int
STAT_CONLOSSMAX	Une perte maximale au héritage des trades déficitaires. La valeur est moins ou égal au zéro	double
STAT_CONLOSSMAX_TRADES	Le nombre de trades qui ont forméSTAT_CONLOSSMAX (une perte maximale au héritage des trades déficitaires)	int
STAT_MAX_CONLOSSES	La perte totale dans une plus longue série des trades déficitaires	double
STAT_MAX_CONLOSS_TRADES	Le nombre de trades dans la plus longue série de trades déficitairesSTAT_MAX_CONLOSSES	int
STAT_BALANCEMIN	La valeur minimale de la balance	double
STAT_BALANCE_DD	Un prélèvement maximal de la balance en moyens monétaires. En train du commerce la balance peut éprouver la multitude de prélèvements, il faut prendre la plus grande signification.	double
STAT_BALANCEDD_PERCENT	Le prélèvement de la balance en pourcentage, qui a été fixé au moment du prélèvement maximal de la balance en moyens monétaires (STAT_BALANCE_DD).	double
STAT_BALANCE_DDREL_PERCENT	Un prélèvement maximal de la balance en pourcentage. En train du commerce la balance	double

© 2000-2015, MetaQuotes Software Corp.

	peut éprouver la multitude de prélèvements, on fixe une valeur relative en pourcentage et une valeur absolue monétaire pour chaque prélèvement. Rend la plus grande valeur	
STAT_BALANCE_DD_RELATIVE	Le prélèvement de la balance en moyens monétaires, qui a été fixé au moment du prélèvement maximal de la balance en pourcentage (STAT_BALANCE_DDREL_PERCENT).	double
STAT_EQUITYMIN	La valeur minimale des moyens propres	double
STAT_EQUITY_DD	Un prélèvement maximal des moyens en moyens monétaires. En train du commerce les moyens peuvent éprouver la multitude de prélèvements, il faut prendre la plus grande signification.	double
STAT_EQUITYDD_PERCENT	>Le prélèvement des moyens en pourcentage, qui a été fixé au moment du prélèvement maximal des moyens en moyens monétaires (STAT_EQUITY_DD).	double
STAT_EQUITY_DDREL_PERCENT	Un prélèvement maximal des moyens en pourcentage. En train du commerce les moyens peuvent éprouver la multitude de prélèvements, dont on fixe une valeur relative du prélèvement en porcentage. Rend la plus grande valeur	double
STAT_EQUITY_DD_RELATIVE	Le prélèvement des moyens en moyens monétaires, qui a été fixé au moment du prélèvement maximal des moyens en pourcentage (STAT_EQUITY_DDREL_PERCENT).	double
STAT_EXPECTED_PAYOFF	L'attente mathématique du gain	double

© 2000-2015, MetaQuotes Software Corp.

STAT_PROFIT_FACTOR	La rentabilité - le rapportSTAT_GROSS_PROFIT/ STAT_GROSS_LOSS. Si STAT_GROSS_LOSS=0, la rentabilité prend la valeurDBL_MAX	double
STAT_RECOVERY_FACTOR	Le facteur de la restitution - le rapportSTAT_PROFIT/ STAT_BALANCE_DD	double
STAT_SHARPE_RATIO	Le coefficient de Scharp	double
STAT_MIN_MARGINLEVEL	La valeur minimale atteinte du niveau de la marge	double
STAT_CUSTOM_ONTESTER	La valeur du critère calculé d'utilisateur de l'optimisation rendu par la fonctionOnTester()	double
STAT_DEALS	Le nombre de marchés faits	int
STAT_TRADES	Le nombre de trades	int
STAT_PROFIT_TRADES	Les trades profitables	int
STAT_LOSS_TRADES	Les trades déficitaires	int
STAT_SHORT_TRADES	Les trades courts	int
STAT_LONG_TRADES	Les trades longues	int
STAT_PROFIT_SHORTTRADES	Les trades courts profitables	int
STAT_PROFIT_LONGTRADES	Les trades longues profitables	int
STAT_PROFITTRADES_AVGCON	Une moyenne longueur de la série profitable des trades	int
STAT_LOSSTRADES_AVGCON	Une moyenne longueur de la série déficitaire des trades	int

© 2000-2015, MetaQuotes Software Corp.

Les constantes commerciales

Les diverses constantes utilisées pour la programmation des stratégies commerciales, sont divisées en groupes suivants:

- L'information sur les données historiques sur l'instrument – la réception de l'information générale selon l'instrument financier;
- Les propriétés des ordres – la réception de l'information sur les ordres commerciaux;
- Les propriétés des positions – la réception de l'information des positions courantes;
- Les propriétés des marchés – la réception de l'information sur les marchés parfaits;
- Les types des transactions commerciales – la description des transactions commerciales accessibles;
- Les types des transactions commerciales -la description des types possibles des transactions commerciales;
- Les ordres commerciaux dans le profondeur de marché - la séparation d'ordres selon la direction d'une opération demandée commerciale.

© 2000-2015, MetaQuotes Software Corp.

L'information sur les données historiques sur l'instrument

A l'accès vers les séries temporelles pour la réception d'une supplémentaire information sur l'instrument on utilise la fonction SeriesInfoInteger(). A titre du paramètre de cette fonction on transmet l'identificateur de la propriété demandée, qui peut accepter une des valeurs d'énumération ENUM_SERIES_INFO_INTEGER.

ENUM_SERIES_INFO_INTEGER

Identificateur	Description	Type de propriété
SERIES_BARS_COUNT	Le nombre de barres selon le symbole-la période pour ce moment	long
SERIES_FIRSTDATE	La première date pour le symbole-la période pour le moment actuel	datetime
SERIES_LASTBAR_DATE	Le temps de l'ouverture de la dernière barre selon le caractère-période	datetime
SERIES_SERVER_FIRSTDATE	La première date à l'histoire selon le symbole sur le serveur indépendamment de la période	datetime
SERIES_TERMINAL_FIRSTDATE	La première date à l'histoire selon le symbole dans le terminal de client indépendamment de la période	datetime
SERIES_SYNCRONIZED	L'indice de la synchronisation des données selon le symbole/ période pour ce moment	bool

© 2000-2015, MetaQuotes Software Corp.

Les propriétés des ordres

Les ordres sur la tenue des transactions commerciales se concrétisent par les ordres. Chaque ordre a la multitude de propriétés pour la lecture, on peut recevoir l'information sur eux à l'aide des fonctions OrderGet...() et HistoryOrderGet...().

Pour les fonctions OrderGetInteger() et HistoryOrderGetInteger()

ENUM_ORDER_PROPERTY_INTEGER

Identificateur	Description	Type
ORDER_TIME_SETUP	Le temps de l'organisation de l'ordre	datetime
ORDER_TYPE	Le type de l'ordre	ENUM_ORDER_TYPE
ORDER_STATE	Le statut de l'ordre	ENUM_ORDER_STATE
ORDER_TIME_EXPIRATION	Le temps de l'expiration de l'ordre	datetime
ORDER_TIME_DONE	Le temps de l'exécution ou le retrait de l'ordre	datetime
ORDER_TIME_SETUP_MSC	Le temps de l'établissement de l'ordre pour l'exécution en millisecondes depuis le 01.01.1970	long
ORDER_TIME_DONE_MSC	Le temps de l'exécution/du retrait de l'ordre en millisecondes depuis le 01.01.1970	long
ORDER_TYPE_FILLING	Le type de l'exécution selon le reste	ENUM_ORDER_TYPE_FILLING
ORDER_TYPE_TIME	Le temps de la vie de l'ordre	ENUM_ORDER_TYPE_TIME
ORDER_MAGIC	L'identificateur de l'expert exposant l'ordre (est destiné pour que chaque expert expose le numéro personnel unique)	long
ORDER_POSITION_ID	L'identificateur de la position, qui est mis sur l'ordre à son exécution. Chaque ordre exécuté engendre le marché, qui ouvreune nouvelle postion ou change la position déjà existante. L'identificateur de cette position s'établit à l'ordre exécuté à ce moment.	long

© 2000-2015, MetaQuotes Software Corp.

Pour les fonctions OrderGetDouble() et HistoryOrderGetDouble()

ENUM_ORDER_PROPERTY_DOUBLE

Identificateur	Description	Type
ORDER_VOLUME_INITIAL	Le volume initial à l'organisation de l'ordre	double
ORDER_VOLUME_CURRENT	Le volume non exécuté	double
ORDER_PRICE_OPEN	Le prix indiqué dans l'ordre	double
ORDER_SL	Le niveau Stop Loss	double
ORDER_TP	Le niveau Take Profit	double
ORDER_PRICE_CURRENT	Le prix actuel selon le symbole de l'ordre	double
ORDER_PRICE_STOPLIMIT	Le prix de l'organisation de l'ordre à cours limité au fonctionnement de l'ordre StopLimit	double

Pour les fonctions OrderGetString() et HistoryOrderGetString()

ENUM_ORDER_PROPERTY_STRING

Identificateur	Description	Type
ORDER_SYMBOL	Le symbole, selon lequel on expose l'ordre	string
ORDER_COMMENT	Le commentaire	string

A l'expédition de la demande commerciale par la fonction OrderSend() pour certaines opérations il est nécessaire d'indiquer le type de l'ordre. Le type de l'ordre est indiqué au champ *type* de la structure spéciale MqlTradeRequest, et peut prendre les valeurs de l' énumération ENUM_ORDER_TYPE.

ENUM_ORDER_TYPE

Identificateur	Description
ORDER_TYPE_BUY	L'ordre de marché sur l'achat
ORDER_TYPE_SELL	L'ordre de marché sur la vente
ORDER_TYPE_BUY_LIMIT	L'ordre remis Buy Limit
ORDER_TYPE_SELL_LIMIT	L'ordre remis Sell Limit
ORDER_TYPE_BUY_STOP	L'ordre remis Buy Stop
ORDER_TYPE_SELL_STOP	L'ordre remis Sell Stop
ORDER_TYPE_BUY_STOP_LIMIT	Pour l'obtention du prix de l'ordre se présente

© 2000-2015, MetaQuotes Software Corp.

	l'ordre remis Buy Limit au prix de StopLimit
ORDER_TYPE_SELL_STOP_LIMIT	Pour l'obtention du prix de l'ordre se présente l'ordre remis Sell Limit au prix de StopLimit

Chaque ordre a le statut décrivant son état. Pour la réception de l'information utilisez la fonction OrderGetInteger() ou HistoryOrderGetInteger() avec le modificateur ORDER_STATE. Les valeurs admissibles se trouvent dans l' énumération ENUM_ORDER_STATE.

ENUM_ORDER_STATE

Identificateur	Description
ORDER_STATE_STARTED	L'ordre est contrôlé sur la convenance, mais n'est pas encore accepté par le broker
ORDER_STATE_PLACED	L'ordre est accepté
ORDER_STATE_CANCELED	L'ordre est retiré par le client
ORDER_STATE_PARTIAL	L'ordre a été exécuté partiellement
ORDER_STATE_FILLED	L'ordre a été exécuté complètement
ORDER_STATE_REJECTED	L'ordre est rejeté
ORDER_STATE_EXPIRED	L'ordre est retiré à l'expiration du terme ses actions
ORDER_STATE_REQUEST_ADD	L'ordre dans l'état de l'enregistrement (l'exposition au système commercial)
ORDER_STATE_REQUEST_MODIFY	L'ordre dans l'état de la modification (le changement de ses paramètres)
ORDER_STATE_REQUEST_CANCEL	L'ordre dans l'état de la suppression (la suppression du système commercial)

A l'expédition de la demande commerciale par la fonction OrderSend() l'ordre peut spécifier la politique de l'exécution au champ *type_filling*dans la structure spéciale MqlTradeRequest, les valeurs sont admissibles d'énumération ENUM_ORDER_TYPE_FILLING. Pour la réception de la valeur de cette propriété utilisez la fonction OrderGetInteger() ou HistoryOrderGetInteger() avec le modificateur ORDER_TYPE_FILLING.

ENUM_ORDER_TYPE_FILLING

Identificateur	La description
ORDER_FILLING_FOK	Cette politique de l'exécution signifie que l'ordre peut être exceptionnellement exécuté dans le volume indiqué. Si au marché au moment donné il n'y a pas de volume suffisant de l'instrument financier, l'ordre ne sera pas

© 2000-2015, MetaQuotes Software Corp.

	exécuté. Le volume nécessaire peut être fait de quelques propositions accessibles au moment donné sur le marché.
ORDER_FILLING_IOC	Signifie l'accord de passer le marché par le volume au maximum accessible sur le marché indiqué à l'ordre. En cas d'impossibilité de l'exécution complète l'ordre sera exécuté sur le volume accessible, et le volume non exécuté de l'ordre sera annulé.
ORDER_FILLING_RETURN	Ce mode est utilisé pour les ordres de marché(ORDER_TYPE_BUY et ORDER_TYPE_SELL), limites et de stop de limite (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT) et seulement en modes "Exécution selon le marché" et "Exécution de bourse". En cas de l'exécution partielle l'ordre de marché ou limité avec le volume résiduel n'est pas remis et continue de fonctionner. Un ordre correspondant limité ORDER_TYPE_BUY_LIMIT/ ORDER_TYPE_SELL_LIMIT avec le type de l'exécution ORDER_FILLING_RETURN sera créé à l'activation pour les ordres ORDER_TYPE_BUY_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT.

On peut donner le délai de validité de l'ordre au champ *type_time* de la structure spéciale MqlTradeRequest à l'expédition de la demande commerciale par la fonction OrderSend(). Les valeurs sont admissibles d'énumération ENUM_ORDER_TYPE_TIME. Pour la réception de la valeur de cette propriété utilisez la fonction OrderGetInteger() ou HistoryOrderGetInteger() avec le modificateur ORDER_TYPE_TIME.

ENUM_ORDER_TYPE_TIME

Identificateur	Description
ORDER_TIME_GTC	L'ordre se trouve au tour jusqu'à ce qu'il soit retiré
ORDER_TIME_DAY	L'ordre fonctionnera seulement pendant le jour commercial courant
ORDER_TIME_SPECIFIED	L'ordre fonctionnera jusqu'à la date de l'expiration
ORDER_TIME_SPECIFIED_DAY	L'ordre sera valable jusqu'à 23:59:59 du jour indiqué. Si ce temps ne tombe pas sur la

© 2000-2015, MetaQuotes Software Corp.

	session commerciale, l'expiration se produira au temps commercial immédiat.

© 2000-2015, MetaQuotes Software Corp.

Les propriétés des positions

Le résultat de l'accomplissement des transactions commerciales sont l'ouverture de la position, le changement de son volume et/ou la direction, ou sa liquidation. Les transactions commerciales sont passées en vertu des ordres, expédiés par la fonction OrderSend() en forme des demandes commerciales Pour chaque instrument (symbole) financier seulement une position ouverte est possible. La position a l'assortiment des propriétés accessibles pour les lectures par les fonctions PositionGet...().

Pour la fonction PositionGetInteger()

ENUM_POSITION_PROPERTY_INTEGER

Identificateur	Description	Type
POSITION_TIME	Le temps de l'ouverture de la position	datetime
POSITION_TIME_MSC	Le temps de l'ouverture de la position en millisecondes depuis le 01.01.1970	long
POSITION_TIME_UPDATE	Le temps du changement de la position en millisecondes depuis le 01.01.1970	long
POSITION_TIME_UPDATE_MSC	Le temps du changement de la position en millisecondes depuis le 01.01.1970	long
POSITION_TYPE	Le type de la position	ENUM_POSITION_TYPE
POSITION_MAGIC	Magic number pour la position (regardes ORDER_MAGIC)	long
POSITION_IDENTIFIER	L'identificateur de la position est un nombre unique, qui est assigné à chaque position ouverte à nouveau, et ne change pas pendant toute sa vie. Le virement de la position ne change pas l'identificateur de la position.	long

Pour la fonction PositionGetDouble()

ENUM_POSITION_PROPERTY_DOUBLE

Identificateur	Description	Type
POSITION_VOLUME	Le volume de la position	double
POSITION_PRICE_OPEN	Le prix de la position	double
POSITION_SL	Le niveau Stop Loss pour une position ouverte	double

© 2000-2015, MetaQuotes Software Corp.

POSITION_TP	Le niveau Take Profit pour une position ouverte	double
POSITION_PRICE_CURRENT	La valeur présente selon le symbole	double
POSITION_SWAP	Le swap accumulé	double
POSITION_PROFIT	Un profit courant	double

Pour la fonction PositionGetString()

ENUM_POSITION_PROPERTY_STRING

Identificateur	Description	Type
POSITION_SYMBOL	Le symbole selon lequel la position est ouverte	string
POSITION_COMMENT	Le commentaire sur la position	string

La direction de la position ouverte (l'achat ou la vente) est définie par la valeur d' énumération ENUM_POSITION_TYPE. Pour la réception du type de la position ouverte utilisez la fonction PositionGetInteger() avec le modificateur POSITION_TYPE.

ENUM_POSITION_TYPE

Identificateur	Description
POSITION_TYPE_BUY	L'achat
POSITION_TYPE_SELL	La vente

© 2000-2015, MetaQuotes Software Corp.

Les propriétés des marchés

Le marché est la réflexion du fait de l'accomplissement de la transaction commerciale en vertu de l'ordre, contenant l'ordre commercial. Chaque commerce est décrit par les propriétés qui permettent d'y obtenir des renseignements. Pour la lecture des valeurs des propriétés on utilise les fonctions de l'aspect HistoryDealGet...(), qui rendent les valeurs des énumérations correspondantes.

Pour la fonction HistoryDealGetInteger()

ENUM_DEAL_PROPERTY_INTEGER

Identificateur	Description	Type
DEAL_ORDER	L'ordre, à la base de quel le marché est effectué	long
DEAL_TIME	Le temps du marché	datetime
DEAL_TIME_MSC	Le temps de l'accomplissement du marché en millisecondes depuis le 01.01.1970	long
DEAL_TYPE	Le type du marché	ENUM_DEAL_TYPE
DEAL_ENTRY	La direction du marché – les entrées au marché, la sortie du marché ou le retournement	ENUM_DEAL_ENTRY
DEAL_MAGIC	Magic number pour la position (regardes ORDER_MAGIC)	long
DEAL_POSITION_ID	L'identificateur de la position, à l'ouverture, au changement ou à la clôture où ce marché a participé. Chaque position possède l'identificateur unique, qui est attribué à tous les marchés, exécutés à l'instrument pendant toute la durée de vie de la position.	long

Pour la fonction HistoryDealGetDouble()

ENUM_DEAL_PROPERTY_DOUBLE

Identificateur	Description	Type
DEAL_VOLUME	Le volume du marché	double
DEAL_PRICE	Le prix du marché	double
DEAL_COMMISSION	La commission du marché	double
DEAL_SWAP	Le swap accumulé à la clôture	double
DEAL_PROFIT	Le résultat financier du	double

© 2000-2015, MetaQuotes Software Corp.

	marché	

Pour la fonction HistoryDealGetString()

ENUM_DEAL_PROPERTY_STRING

Identificateur	Description	Type
DEAL_SYMBOL	Le nom du symbole, selon lequel le marché est produit	string
DEAL_COMMENT	Le commentaire au le marché	string

Chaque marché se caractérise par le type, les valeurs possibles sont énumérées à ENUM_DEAL_TYPE. Pour la réception de l'information sur le type du marché utilisez la fonction HistoryDealGetInteger() avec le modificateur DEAL_TYPE.

ENUM_DEAL_TYPE

Identificateur	Description
DEAL_TYPE_BUY	L'achat
DEAL_TYPE_SELL	La vente
DEAL_TYPE_BALANCE	La mise en compte de la balance
DEAL_TYPE_CREDIT	La mise en compte du crédit
DEAL_TYPE_CHARGE	Les taxes supplémentaires
DEAL_TYPE_CORRECTION	L'enregistrement corrigeant
DEAL_TYPE_BONUS	L'énumération des bonus
DEAL_TYPE_COMMISSION	Les commissions supplémentaires
DEAL_TYPE_COMMISSION_DAILY	La commission calculée à la fin du jour commercial
DEAL_TYPE_COMMISSION_MONTHLY	La commission calculée à la fin du mois
DEAL_TYPE_AGENT_DAILY	La commission d'agent calculée à la fin du jour commercial
DEAL_TYPE_AGENT_MONTHLY	La commission d'agent calculée à la fin du mois
DEAL_TYPE_INTERESTRATE	Les mises en compte des pour-cent aux moyens libres
DEAL_TYPE_BUY_CANCELED	Le marché annulé de l'achat. La situation est possible, quand le marché sur l'achat auparavant fait est annulé. Dans ce cas, le type du marché auparavant fait (DEAL_TYPE_BUY) se change sur DEAL_TYPE_BUY_CANCELED, et son bénéfice/perte est remise à zéro. Le bénéfice/perte auparavant reçu est calculée/annulé du

© 2000-2015, MetaQuotes Software Corp.

	compte par l'opération distincte de balance
DEAL_TYPE_SELL_CANCELED	Le marché annulé de la vente. La situation est possible, quand le marché sur la vente auparavant fait est annulé. Dans ce cas, le type du marché auparavant fait (DEAL_TYPE_SELL) se change sur DEAL_TYPE_SELL_CANCELED, et son bénéfice/perte est remise à zéro. Le bénéfice/perte auparavant reçu est calculée/ annulé du compte par l'opération distincte de balance

Les marchés se distinguent non seulement d'après le type, spécifié dans les énumérations ENUM_DEAL_TYPE, mais aussi selon le moyen du changement de la position. Cela peut être une simple ouverture de position ou l'accumulation du volume avant la position ouverte (l'entrée au marché), la clôture de la position par le marché de la direction contraire par le volume correspondant (la sortie de leur marché) ou l'inversion de position dans le cas où le volume du marché couvre le volume de la position auparavant ouverte.

Toutes ces situations sont décrites par les valeurs de l'énumération ENUM_DEAL_ENTRY. Pour la réception de cette information sur le marché utilisez la fonction HistoryDealGetInteger() avec le modificateur DEAL_ENTRY.

ENUM_DEAL_ENTRY

Identificateur	Description
DEAL_ENTRY_IN	L'entrée au marché
DEAL_ENTRY_OUT	La sortie du marché
DEAL_ENTRY_INOUT	L'inversion

© 2000-2015, MetaQuotes Software Corp.

Les types des transactions commerciales

Le commerce se réalise au moyen de l'expédition à l'aide de la fonction OrderSend() ordres sur l'ouverture des positions, ainsi que les ordres sur l'installation, la modification et l'éloignement des ordres remis. Chaque ordre commercial contient l'indication au type de la transaction commerciale demandée. Les transactions commerciales sont décrites dans l'énumération ENUM_TRADE_REQUEST_ACTIONS.

ENUM_TRADE_REQUEST_ACTIONS

Identificateur	Description
TRADE_ACTION_DEAL	Placer l'ordre commercial pour l'exécution immédiate avec les paramètres indiqués (l'ordre du marché)
TRADE_ACTION_PENDING	Placer l'ordre commercial pour l'exécution sous les conditions indiquées (l'ordre remis)
TRADE_ACTION_SLTP	Changer les valeurs Stop Loss et Take Profit de la position ouverte
TRADE_ACTION_MODIFY	Modifiez les paramètres de l'ordre placé auparavant
TRADE_ACTION_REMOVE	Supprimer l'ordrecommercial auparavant remis

© 2000-2015, MetaQuotes Software Corp.

Les types des transactions commerciales

A la suite de l'exécution des actions définies avec le compte commercial, son état change. Ces actions comprennent:

- L'envoi de la demande commerciale par n'importe quelle application MQL5- dans le terminal de client à l'aide des fonctions OrderSend et OrderSendAsync et son exécution ultérieure;
- L'envoi de la demande commerciale par l'interface graphique du terminal et son exécution ultérieure;
- le fonctionnement des ordres remis et des ordres stop sur le serveur;
- L'exécution des opérations sur le côté du serveur commercial.

A la suite de ces actions, les transactions commerciales sont exécutées pour le compte:

- le traitement de la demande commerciale;
- le changement des ordres ouverts;
- le changement de l'histoire des ordres;
- le changement de l'histoire des marchés;
- le changement des positions.

Par exemple, à l'envoi de l'ordre de marché sur l'achat, il est traité, l'ordre correspondant sur l'achat est créé pour le compte, se passe l'exécution de l'ordre, son suppression de la liste des ouverts, l'ajout à l'histoire des ordres, ensuite le marché correspondant est ajouté à l'histoire et une nouvelle position est créé. Toutes ces actions sont les transactions commerciales.

Pour que le programmeur puisse suivre les actions réalisées relativement au compte commercial, il y a la fonction OnTradeTransaction. A l'aide de ce gestionnaire dans l'application MQL5 on peut recevoir les transactions commerciales appliquées au compte. La description de la transaction commerciale est transmise dans un premier paramètre OnTradeTransaction à l'aide de la structure MqlTradeTransaction.

Le type de la transaction commerciale est transmis dans le paramètre type de la structure MqlTradeTransaction. Les types possibles des transactions commerciales sont décrits par l'énumération suivante:

ENUM_TRADE_TRANSACTION_TYPE

Identificateur	La description
TRADE_TRANSACTION_ORDER_ADD	L'ajout d'un nouvel ordre ouvert.
TRADE_TRANSACTION_ORDER_UPDATE	Le changement de l'ordre ouvert. Non seulement les changements évidents du côté du terminal de client ou du serveur commercial se rapportent aux changements donnés, mais aussi le changement de son état à son exposition (par exemple, le passage de l'étatORDER_STATE_STARTED au ORDER_STATE_PLACED ou de ORDER_STATE_PLACED au ORDER_STATE_PARTIAL etc.).
TRADE_TRANSACTION_ORDER_DELETE	La suppression de l'ordre de la liste des ouverts.

© 2000-2015, MetaQuotes Software Corp.

	L'ordre peut être supprimé des ouverts à la suite de l'exposition de la demande correspondante ou à la suite de l'exécution (du remplissage) et le transfert dans l'histoire.
TRADE_TRANSACTION_DEAL_ADD	L'ajout du marché dans l'histoire. Se réalise à la suite de l'exécution de l'ordre ou de la tenue des opérations avec la balance du compte.
TRADE_TRANSACTION_DEAL_UPDATE	Le changement du marché dans l'histoire. Les situations sont possibles, quand le marché auparavant exécuté se change sur le serveur. Par exemple, le marché était changé dans le système (la bourse) extérieur commercial, où il était déduit par le broker.
TRADE_TRANSACTION_DEAL_DELETE	La suppression du marché de l'histoire. Les situations sont possibles, quand le marché auparavant exécuté est supprimé sur le serveur. Par exemple, le marché était supprimé dans le système (la bourse) extérieur commercial, où il était déduit par le broker.
TRADE_TRANSACTION_HISTORY_ADD	L'ajout de l'ordre dans l'histoire à la suite de l'exécution ou de la suppression.
TRADE_TRANSACTION_HISTORY_UPDATE	Le changement de l'ordre qui se trouve dans l'histoire des ordres. Ce type est prévu pour l'élargissement de la fonctionnalité sur le côté du serveur commercial.
TRADE_TRANSACTION_HISTORY_DELETE	La suppression de l'ordre de l'histoire des ordres. Ce type est prévu pour l'élargissement de la fonctionnalité sur le côté du serveur commercial.
TRADE_TRANSACTION_POSITION	Le changement de la position non lié à l'exécution du marché. Ce type de la transaction témoigne notamment de ce que la position était changée sur le côté du serveur commercial. Le volume, le prix de l'ouverture, ainsi que les niveaux Stop Loss et Take Profit peuvent être changés dans la position. L'information sur les changements est transmise dans la structure MqlTradeTransaction par le gestionnaire OnTradeTransaction. Le changement de la position (l'ajout, le changement ou la liquidation) à la suite de l'exécution du marché n'entraîne pas l'apparition de la transaction TRADE_TRANSACTION_POSITION.
TRADE_TRANSACTION_REQUEST	La notification que la demande commerciale est traitée par le serveur, et le résultat de son traitement est reçu. Pour les transaction de ce

© 2000-2015, MetaQuotes Software Corp.

	type dans la structureMqlTradeTransaction il faut analyser seulement un champ - type (le type de la transaction). Pour la réception de l'information supplémentaire il est nécessaire d'analyser les deuxièmes et troisièmes paramètres de la fonction OnTradeTransaction (request et result).

En fonction du type de la transaction commerciale, dans la structure MqlTradeTransaction, qui la décrit, les différents paramètres sont remplis. La description détaillée des données transmises est amenée dans le paragraphe"La structure de la transaction commerciale".

Voir aussi

La structure de la transaction commerciale, OnTradeTransaction

© 2000-2015, MetaQuotes Software Corp.

Les ordres commerciaux dans le profondeur de marché

Pour les faits sur instruments boursiers la fenêtre est accessible "le tour des demandes", où on peut regarder les demandes courantes pour l'achat et la vente. Pour chaque demande on indique la direction souhaitée de l'opération commerciale, le volume demandé et le prix demandé.

Pour la réception de l'information sur l'état courant du profondeur de marché par les moyens du langage MQL5 est destinée la fonction MarketBookGet(), qui place "la copie d'écran du profondeur du marché" au tableau des structures MqlBookInfo. Chaque élément de ce tableau du champ*type* contient l'information sur la direction de la demande – c'est la valeur de l'énumération ENUM_BOOK_TYPE.

ENUM_BOOK_TYPE

Identificateur	Description
BOOK_TYPE_SELL	La demande pour la vente
BOOK_TYPE_BUY	La demande pour l'achat
BOOK_TYPE_SELL_MARKET	La demande pour la vente au prix du marché
BOOK_TYPE_BUY_MARKET	La demande pour l'achat au prix du marché

Voir aussi

Les structures et les classes, La structure du profondeur de marché, Les types des opérations commerciales, la Réception de l'information de marché

© 2000-2015, MetaQuotes Software Corp.

Свойства сигналов

Значения перечислений для работы с торговыми сигналами и настройками их копирования.

Перечисления свойств типа double торговых сигналов:

ENUM_SIGNAL_BASE_DOUBLE

Константа	Описание
SIGNAL_BASE_BALANCE	Баланс счета
SIGNAL_BASE_EQUITY	Средства на счете
SIGNAL_BASE_GAIN	Прирост счета в процентах
SIGNAL_BASE_MAX_DRAWDOWN	Максимальная просадка
SIGNAL_BASE_PRICE	Цена подписки на сигнал
SIGNAL_BASE_ROI	Значение ROI (Return on Investment) сигнала в %

Перечисления свойств типа integer торговых сигналов:

ENUM_SIGNAL_BASE_INTEGER

Константа	Описание
SIGNAL_BASE_DATE_PUBLISHED	Дата публикации сигнала (когда стал доступен для подписки)
SIGNAL_BASE_DATE_STARTED	Дата начала мониторинга сигнала
SIGNAL_BASE_ID	ID сигнала
SIGNAL_BASE_LEVERAGE	Плечо торгового счета
SIGNAL_BASE_PIPS	Результат торговли в пипсах
SIGNAL_BASE_RATING	Позиция в рейтинге сигналов
SIGNAL_BASE_SUBSCRIBERS	Количество подписчиков
SIGNAL_BASE_TRADES	Количество трейдов
SIGNAL_BASE_TRADE_MODE	Тип счета (0-реальный счет, 1-демо-счет, 2-конкурсный счет)

Перечисления свойств типа string торговых сигналов:

ENUM_SIGNAL_BASE_STRING

Константа	Описание
SIGNAL_BASE_AUTHOR_LOGIN	Логин автора сигнала
SIGNAL_BASE_BROKER	Наименование брокера (компании)

© 2000-2015, MetaQuotes Software Corp.

SIGNAL_BASE_BROKER_SERVER	Сервер брокера
SIGNAL_BASE_NAME	Имя сигнала
SIGNAL_BASE_CURRENCY	Валюта счета сигнала

Перечисления свойств типа double настроек копирования торговых сигналов:

ENUM_SIGNAL_INFO_DOUBLE

Константа	Описание
SIGNAL_INFO_EQUITY_LIMIT	Процент для конвертации объема сделки
SIGNAL_INFO_SLIPPAGE	Величина проскальзывания, с которым выставляются рыночные ордера при синхронизации позиций и копировании сделок
SIGNAL_INFO_VOLUME_PERCENT	Значение ограничения по средствам для сигнала, r/o

Перечисления свойств типа integer настроек копирования торговых сигналов:

ENUM_SIGNAL_INFO_INTEGER

Константа	Описание
SIGNAL_INFO_CONFIRMATIONS_DISABLED	Флаг разрешения синхронизации без показа диалога подтверждения
SIGNAL_INFO_COPY_SLTP	Флаг копирования Stop Loss и Take Profit
SIGNAL_INFO_DEPOSIT_PERCENT	Ограничения по депозиту (в %)
SIGNAL_INFO_ID	id сигнала, r/o
SIGNAL_INFO_SUBSCRIPTION_ENABLED	Флаг разрешения на копирование сделок по подписке
SIGNAL_INFO_TERMS_AGREE	Флаг согласия с условиями использования сервиса "Сигналы", r/o

Перечисления свойств типа string настроек копирования торговых сигналов:

ENUM_SIGNAL_INFO_STRING

Константа	Описание
SIGNAL_INFO_NAME	Имя сигнала, r/o

Смотри также

Управление сигналами

© 2000-2015, MetaQuotes Software Corp.

Les constantes nommées

On peut diviser toutes les constantes utilisées dans le langage MQL5 aux groupes suivants:

- Les macrosubstitutions prédéterminées - les valeurs sont substituées pendant la compilation;
- Les constantes mathématiques - les valeurs de certaines expressions mathématiques;
- Les constantes des types numériques- certaines des limitations des types simples;
- Les raisons de la déinitialisation - la description des raisons de la déinitialisation;
- Le contrôle de l'indicateur de l'objet - l'énumération des types des indicateurs, rendus par la fonction CheckPointer() ;
- Les autres constantes - tout que ne faisait pas partie des autres groupes des constantes.

© 2000-2015, MetaQuotes Software Corp.

Les macrosubstitutions prédéterminées

Pour l'allégement du réglage et la réception de l'information sur le travail du programme mql5 on introduit les constantes-macros spéciales, dont les valeurs s'établient au moment de la compilation. Le chemin le plus simple de l'utilisation de ces constantes - la sortie des valeurs à l'aide de la fonction Print(), comme c'est montré dans l'exemple.

Constante	Description
__DATE__	La date de la compilation du fichier sans l'heure (les heures, les minutes et les secondes sont égales 0)
__DATETIME__	La date et le temps de la compilation du fichier
__LINE__	Le numéro de la ligne dans le code source, où se trouve la macro donnée
__FILE__	Le nom du fichier courant compilé
__PATH__	Le chemin absolu vers le fichier compilé actuel
__FUNCTION__	Le nom de la fonction, dans le corps duquel se trouve la macro
__FUNCSIG__	La signature de la fonction, dont dans le corps la macro est située. La sortie de la description complète de la fonction avec les types des paramètres dans le log peut être utile à l'identification des fonctions surchargées
__MQLBUILD__, __MQL5BUILD__	Le numéro du build du compilateur

Exemple:

```
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "http://www.metaquotes.net"
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- exemple de la sortie de l'information à l'initialisation du conseiller
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//--- mettre l'intervalle entre les événements de minuteur
   EventSetTimer(5);
//---
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- exemple de la sortie de l'information à la déinitialisation du conseiller
```

© 2000-2015, MetaQuotes Software Corp.

```
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//---
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {

//--- sortie de l'information à l'entrée du tick
   Print(" __MQLBUILD__ = ",__MQLBUILD__,"  __FILE__ = ",__FILE__);
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
   test2();
//---
  }
//+------------------------------------------------------------------+
//| test1                                                            |
//+------------------------------------------------------------------+
void test1(string par)
  {
//--- la sortie de l'information à l'intérieur de la fonction
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__," par=",par);
  }
//+------------------------------------------------------------------+
//| test2                                                            |
//+------------------------------------------------------------------+
void test2()
  {
//--- sortie de l'information à l'intérieur de la fonction
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
  }
//+------------------------------------------------------------------+
//| OnTimer event handler                                            |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Les constantes mathématiques

Pour certaines expressions mathématiques on réserve les constantes spéciales contenant les valeurs. On peut utiliser ces constantes dans n'importe quel espace du programme mql5 au lieu du calcul de leur valeur à l'aide des fonctions mathématiques.

Constante	Description	Valeur
M_E	e	2.71828182845904523536
M_LOG2E	log2(e)	1.44269504088896340736
M_LOG10E	log10(e)	0.43429448190325182765 1
M_LN2	ln(2)	0.69314718055994530941 7
M_LN10	ln(10)	2.30258509299404568402
M_PI	pi	3.14159265358979323846
M_PI_2	pi/2	1.57079632679489661923
M_PI_4	pi/4	0.78539816339744830962
M_1_PI	1/pi	0.31830988618379067153 8
M_2_PI	2/pi	0.63661977236758134307 6
M_2_SQRTPI	2/sqrt(pi)	1.12837916709551257390
M_SQRT2	sqrt(2)	1.41421356237309504880
M_SQRT1_2	1/sqrt(2)	0.70710678118654752440 1

Exemple:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- sortons les valeurs des constantes
   Print("M_E = ",DoubleToString(M_E,16));
   Print("M_LOG2E = ",DoubleToString(M_LOG2E,16));
   Print("M_LOG10E = ",DoubleToString(M_LOG10E,16));
   Print("M_LN2 = ",DoubleToString(M_LN2,16));
   Print("M_LN10 = ",DoubleToString(M_LN10,16));
   Print("M_PI = ",DoubleToString(M_PI,16));
   Print("M_PI_2 = ",DoubleToString(M_PI_2,16));
   Print("M_PI_4 = ",DoubleToString(M_PI_4,16));
   Print("M_1_PI = ",DoubleToString(M_1_PI,16));
   Print("M_2_PI = ",DoubleToString(M_2_PI,16));
   Print("M_2_SQRTPI = ",DoubleToString(M_2_SQRTPI,16));
   Print("M_SQRT2 = ",DoubleToString(M_SQRT2,16));
   Print("M_SQRT1_2 = ",DoubleToString(M_SQRT1_2,16));
```

© 2000-2015, MetaQuotes Software Corp.

```
}
```

© 2000-2015, MetaQuotes Software Corp.

Les constantes des types de données numériques

Chaque type simple numérique est destiné au cercle défini des tâches et permet d'optimiser le travail du programme mql5 à l'application juste. Pour la meilleure lisibilité du code et le traitement juste des résultats des calculs on introduit les constantes, qui permettent de recevoir l'information sur les bornages imposés à n'importe quel type des données simples.

Constante	Description	Valeur
CHAR_MIN	La valeur minimale, qui peut être présentée par le type char	-128
CHAR_MAX	La valeur maximale, qui peut être présentée par le type char	127
UCHAR_MAX	La valeur maximale, qui peut être présentée par le type uchar	255
SHORT_MIN	La valeur minimale, qui peut être présentée par le type short	-32768
SHORT_MAX	La valeur maximale, qui peut être présentée par le type short	32767
USHORT_MAX	La valeur maximale, qui peut être présentée par le type ushort	65535
INT_MIN	La valeur minimale, qui peut être présentée par le type int	-2147483648
INT_MAX	La valeur maximale, qui peut être présentée par le type int	2147483647
UINT_MAX	La valeur maximale, qui peut être présentée par le type uint	4294967295
LONG_MIN	La valeur minimale, qui peut être présentée par le type long	-9223372036854775808
LONG_MAX	La valeur maximale, qui peut être présentée par le type long	9223372036854775807
ULONG_MAX	La valeur maximale, qui peut être présentée par le type ulong	18446744073709551615
DBL_MIN	La valeur minimale positive, qui peut être présentée par le type double	2.2250738585072014e-308
DBL_MAX	La valeur maximale, qui peut être présentée par le type double	1.7976931348623158e+308

© 2000-2015, MetaQuotes Software Corp.

DBL_EPSILON	La valeur minimale, à qui satisfait la condition : 1.0 +DBL_EPSILON != 1.0	2.2204460492503131e-016
DBL_DIG	La quantité de signes signifiants décimaux	15
DBL_MANT_DIG	La quantité de bits dans la mantisse	53
DBL_MAX_10_EXP	La valeur maximale décimale du degré d'exposant	308
DBL_MAX_EXP	La valeur maximale binaire du degré d'exposant	1024
DBL_MIN_10_EXP	La valeur minimale décimale du degré d'exposant	(-307)
DBL_MIN_EXP	La valeur minimale binaire du degré d'exposant	(-1021)
FLT_MIN	La valeur minimale positive, qui peut être présentée par le type float	1.175494351e-38
FLT_MAX	La valeur maximale, qui peut être présentée par le type float	3.402823466e+38
FLT_EPSILON	La valeur minimale, qui satisfait à la condition : 1.0 +FLT_EPSILON != 1.0	1.192092896e-07
FLT_DIG	Le nombre de signes signifiants décimaux	6
FLT_MANT_DIG	Le nombre de bits dans la mantisse	24
FLT_MAX_10_EXP	La valeur maximale décimale du degré d'exposant	38
FLT_MAX_EXP	La valeur maximale binaire du degré d'exposant	128
FLT_MIN_10_EXP	La valeur minimale décimale du degré d'exposant	-37
FLT_MIN_EXP	La valeur minimale binaire du degré d'exposant	(-125)

Exemple:

```
void OnStart()
  {
//--- sortons les valeurs des constantes
   printf("CHAR_MIN = %d",CHAR_MIN);
```

© 2000-2015, MetaQuotes Software Corp.

```
   printf("CHAR_MAX = %d",CHAR_MAX);
   printf("UCHAR_MAX = %d",UCHAR_MAX);
   printf("SHORT_MIN = %d",SHORT_MIN);
   printf("SHORT_MAX = %d",SHORT_MAX);
   printf("USHORT_MAX = %d",USHORT_MAX);
   printf("INT_MIN = %d",INT_MIN);
   printf("INT_MAX = %d",INT_MAX);
   printf("UINT_MAX = %u",UINT_MAX);
   printf("LONG_MIN = %I64d",LONG_MIN);
   printf("LONG_MAX = %I64d",LONG_MAX);
   printf("ULONG_MAX = %I64u",ULONG_MAX);
   printf("EMPTY_VALUE = %.16e",EMPTY_VALUE);
   printf("DBL_MIN = %.16e",DBL_MIN);
   printf("DBL_MAX = %.16e",DBL_MAX);
   printf("DBL_EPSILON = %.16e",DBL_EPSILON);
   printf("DBL_DIG = %d",DBL_DIG);
   printf("DBL_MANT_DIG = %d",DBL_MANT_DIG);
   printf("DBL_MAX_10_EXP =  %d",DBL_MAX_10_EXP);
   printf("DBL_MAX_EXP = %d",DBL_MAX_EXP);
   printf("DBL_MIN_10_EXP = %d",DBL_MIN_10_EXP);
   printf("DBL_MIN_EXP = %d",DBL_MIN_EXP);
   printf("FLT_MIN = %.8e",FLT_MIN);
   printf("FLT_MAX = %.8e",FLT_MAX);
   printf("FLT_EPSILON = %.8e",FLT_EPSILON);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Les raisons de la déinitialisation

Les codes de la raison de la déinitialisation de l'expert, récupéré par la fonction UninitializeReason(). Ils peuvent avoir chacune des valeurs suivantes:

Constante	Valeur	Description
REASON_PROGRAM	0	L'expert a cessé le travail, ayant provoqué la fonction ExpertRemove()
REASON_REMOVE	1	Le programme est supprimé du graphique
REASON_RECOMPILE	2	Le programme a été recompilé
REASON_CHARTCHANGE	3	Le caractère ou la période du graphique était changé
REASON_CHARTCLOSE	4	Le graphique est fermé
REASON_PARAMETERS	5	Les paramètres d'entrée étaient changés par l'utilisateur
REASON_ACCOUNT	6	Un autre compte a été activé ou la reconnexion vers le serveur commercial est produite à la suite de changements dans les paramètres du compte
REASON_TEMPLATE	7	Un autre modèle du graphique a été appliqué
REASON_INITFAILED	8	Cette valeur signifit que OnInit() a rendu la valeur non nulle
REASON_CLOSE	9	Le terminal a été fermé

Le code de la raison de la déinitialisation est transmis aussi à titre du paramètre de la fonction prédéterminée OnDeinit(const int reason).

Les indicateurs acceptent seulement le code 1(REASON_REMOVE) et le code 2(REASON_RECOMPILE).

Exemple:

```
//+------------------------------------------------------------------+
//| get text description                                             |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
```

© 2000-2015, MetaQuotes Software Corp.

```
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="Account was changed";break;
      case REASON_CHARTCHANGE:
         text="Symbol or timeframe was changed";break;
      case REASON_CHARTCLOSE:
         text="Chart was closed";break;
      case REASON_PARAMETERS:
         text="Input-parameter was changed";break;
      case REASON_RECOMPILE:
         text="Program "+__FILE__+" was recompiled";break;
      case REASON_REMOVE:
         text="Program "+__FILE__+" was removed from chart";break;
      case REASON_TEMPLATE:
         text="New template was applied to chart";break;
      default:text="Another reason";
     }
//---
   return text;
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- premier moyen de recevoir le code de la raison de la déinitialisation
   Print(__FUNCTION__,"Le code de la raison de la déinitialisation  = ",reason);
//--- deuxième moyen de recevoir le code de la raison de la déinitialisation
   Print(__FUNCTION__,"_UninitReason = ",getUninitReasonText(_UninitReason));
  }
```

© 2000-2015, MetaQuotes Software Corp.

Le contrôle de l'indicateur de l'objet

Pour le contrôle du type de l'indicateur de l'objet est destiné la fonction CheckPointer(), qui rend la valeur de l'énumération ENUM_POINTER_TYPE. En cas de l'utilisation de l'indicateur incorrect l'exécution du programme sera immédiatement interrompue.

Les objets créés par l'opérateur new, ont le type POINTER_DYNAMIC. Pour tels indicateurs il faut utiliser l'opérateur de l'annulation delete().

Tous les autres indicateurs ont le type POINTER_AUTOMATIC, qui signifie que l'objet donné était automatiquement créé par l'environnement de l'exécution du programme mql5. Tels objets sont supprimés automatiquement après l'utilisation.

ENUM_POINTER_TYPE

Constante	Description
POINTER_INVALID	L'indicateur incorrect
POINTER_DYNAMIC	L'indicateur de l'objet créé par l'opérateur new
POINTER_AUTOMATIC	L'indicateur de n'importe quel objet créé automatiquement (sans utilisation new())

Voir aussi

Les erreurs de l'exécution, l'opérateur de l'annulation de l'objet delete, CheckPointer()

© 2000-2015, MetaQuotes Software Corp.

Les autres constantes

La constante CLR_NONE sert pour l'indication de l'absence de la couleur, c'est-à-dire l'objet graphique ou une série graphique de l'indicateur ne seront pas affichés. Il n' y a pas de cette constante dans la liste des constantes avec les noms des couleurs Web, mais elle peut être appliqué partout, où on demande l'indication de la couleur.

La constante INVALID_HANDLE peut être utilisée à la vérification des handles du fichier (regardes les fonctions FileOpen() et FileFindFirst()).

Constante	Description	Valeur
CHARTS_MAX	Le nombre maximale possible des graphiques ouverts ensemble dans le terminal	100
clrNONE	L'absence de la couleur	-1
EMPTY_VALUE	La valeur vide dans le tampon d'indicateur	DBL_MAX
INVALID_HANDLE	Le handle incorrect	-1
IS_DEBUG_MODE	Le critère du travail du programme mq5 en mode du débogage	true en mode du réglage, dans le cas contraire c'est false
IS_PROFILE_MODE	Le critère du travail du programme mq5 en mode du profilage	en mode du profilage n'est pas égal au zéro, dans le cas contraire 0
NULL	Le zéro de n'importe quel type	0
WHOLE_ARRAY	Signifie le nombre d'éléments restés jusqu'à la fin du tableau, c'est-à-dire, tout le tableau sera traité	-1
WRONG_VALUE	La constante peut non évidemment être amenée au type de n'importe quelle énumération	-1

La constante EMPTY_VALUE correspond généralement à la valeur des indicateurs qui ne sont pas indiquées sur le graphique. Par exemple, pour l'indicateur inséré Standard Deviation la ligne pour les premiers 19 barres dans l'histoire n'est pas visualisé sur le graphique. Si créer le handle de cet indicateur à l'aide de la fonction iStdDev() et copier au tableau la valeur de l'indicateur pour ces barres par CopyBuffer(), ces valeurs seront égales EMPTY_VALUE.

On peut indiquer de soi-même dans l'indicateur d'utilisateur la valeur personnelle vide de l'indicateur, à laquelle ne doit pas être produite le dessin sur le graphique. Pour cela utilisez la fonction PlotIndexSetDouble() avec le modificateur PLOT_EMPTY_VALUE.

La constante NULL peut être assignée par la variable de n'importe quel type simple ou l'indicateur sur l'objet de la structure ou la classe. L'assignement NULL de la variable de ligne signifie une

© 2000-2015, MetaQuotes Software Corp.

déinitialisation complète de cette variable.

La constante WRONG_VALUE est destinée à ces cas, quand il faut rendre la valeur de l'énumération, et ce doit être la valeur incorrecte. Par exemple, il faut informer, la valeur récupérée est la valeur de cet énumération. Prenons comme une illustration une certaine fonction CheckLineStyle(), qui rend le style de la ligne pour l'objet indiqué du nom. Si à la requête du style par la fonction ObjectGetInteger()le résultat sera true, reviendra la valeur d'énumération ENUM_LINE_STYLE, autrement revient WRONG_VALUE.

```
void OnStart()
  {
   if(CheckLineStyle("MyChartObject")==WRONG_VALUE)
      printf("Error line style getting.");
  }
//+------------------------------------------------------------------+
//| Rend le style de la ligne pour l'objet indiqué du nom            |
//+------------------------------------------------------------------+
ENUM_LINE_STYLE CheckLineStyle(string name)
  {
   long style;
//---
   if(ObjectGetInteger(0,name,OBJPROP_STYLE,0,style))
      return((ENUM_LINE_STYLE)style);
   else
      return(WRONG_VALUE);
  }
```

La constante WHOLE_ARRAY est destinée aux fonctions, qui demandent l'indication de la quantité d'éléments dans les tableaux traités:

- ArrayCopy();
- ArrayMinimum();
- ArrayMaximum();
- FileReadArray();
- FileWriteArray().

S'il faut indiquer qu'il est nécessaire de traiter toutes les valeurs du tableaux de la position indiquée et jusqu'à la fin, il suffit d'indiquer la valeur WHOLE_ARRAY.

La constanteIS_PROFILE_MODE permet de changer le travail du programme pour la recherche de l'information correcte en mode du profilage. Le profilage permet de mesurer le temps de l'exécution des fragments séparés du programme (d'habitude ce sont les fonctions), ainsi que compter la quantité de tels appels. Pour la réception correcte de l'information sur le temps de l'exécution en mode du profilage on peut désactiver les appels de la fonction Sleep() comme dans l'exemple

```
//--- Sleep peut fortement influencer (contrefaire) sur le résultat du profilage
if(!IS_PROFILE_MODE) Sleep(100); // interdisons l'appel Sleep() en mode du profilage
```

La valeur de la constante IS_PROFILE_MODE est spécifiée par le compilateur au moment de la compilation, et en mode ordinaire cette valeur se présente comme zéro. Au démarrage du programme en mode du profilage on produit la compilation spéciale, et dans ce cas au lieu de IS_PROFILE_MODE la valeur différente de zéro est substituée.

La constante IS_DEBUG_MODE sera utile dans les cas où il est nécessaire un peu de changer le travail

© 2000-2015, MetaQuotes Software Corp.

du programme mql5 en mode du débogage. Par exemple, en mode du débogage peut être nécessaire de visualiser l'information supplémentaire de réglage à log du terminal ou créer les objets auxiliaires graphiques sur le graphique.

L'exemple cité ci-dessous crée l'objet Label et donne sa description et la couleur en fonction de celui-là, en quel mode on accomplit le script. Pour lancer le script en mode du débogage de MetaEditor, appuyez sur la touche **F5**. Si lancer du script de la fenêtre du navigateur dans le terminal, la couleur et le texte de l'objet Label seront les autres.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Check_DEBUG_MODE.mq5 |
//|                        Copyright © 2009, MetaQuotes Software Corp. |
//|                                             http://www.metaquotes.net |
//+------------------------------------------------------------------+
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "http://www.metaquotes.net"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="invisible_label";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Object ",label_name," not found. Error code = ",GetLastError());
      //--- créons l'objet Label
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- établissons la coordonnée X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- établissons la coordonnée Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      ResetLastError();
      if(IS_DEBUG_MODE) // mode du débogage
        {
         //--- visualiserons le message sur le mode de l'exécution du script
         ObjectSetString(0,label_name,OBJPROP_TEXT,"DEBUG MODE");
         //--- spécifions la couleur rouge du texte
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrRed))
            Print("On n'a pas réussi à établir la couleur.L'erreur ",GetLastError());
        }
      else              // mode opérationnel
        {
         ObjectSetString(0,label_name,OBJPROP_TEXT,"RELEASE MODE");
         //--- spécifions la couleur invisible du texte
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,CLR_NONE))
            Print("On n'a pas réussi à établir la couleur.L'erreur ",GetLastError());
        }
      ChartRedraw();
      DebugBreak();     // Il y aura ici une interruption, si nous sommes en mode du dé
     }
  }
```

Методы шифрования данных

© 2000-2015, MetaQuotes Software Corp.

Для указания метода преобразования данных (шифрование и расчет хешей) в функциях CryptEncode() и CryptDecode() используется перечисление ENUM_CRYPT_METHOD.

ENUM_CRYPT_METHOD

Константа	Описание
CRYPT_BASE64	Шифрование BASE64 (перекодировка)
CRYPT_AES128	Шифрование AES с ключом 128 бит (16 байт)
CRYPT_AES256	Шифрование AES с ключом 256 бит (32 байта)
CRYPT_DES	Шифрование DES с ключом 56 бит (7 байт)
CRYPT_HASH_SHA1	Расчёт HASH SHA1
CRYPT_HASH_SHA256	Расчёт HASH SHA256
CRYPT_HASH_MD5	Расчёт HASH MD5
CRYPT_ARCH_ZIP	ZIP архивирование

Voir aussi

DebugBreak, l'information sur le programme MQL5 exécuté, CryptEncode(), CryptDecode()

© 2000-2015, MetaQuotes Software Corp.

Les structures des données

À MQL5 il y a les 8 structures prédéterminées destinées au stockage et la transmission de l'information de service:

- MqlDateTime est destinée à la représentation de la date et le temps;
- MqlParam permet de transmettre les paramètres d'entrée à la création du handle de l'indicateur avec l'aide de la fonction IndicatorCreate();
- MqlRates accorde l'information sur les données historiques , contenant le prix, le volume et le spread;
- MqlBookInfo pour la réception de l'information affichée dans le profondeur de marché (la fenêtre des cotations);
- MqlTradeRequest pour la création de la requête commerciale à la tenue des opérations commerciales;
- MqlTradeCheckResult permet de vérifier la demande commerciale préparée avant son envoi;
- MqlTradeResult contient la réponse du serveur commercial à la requête commerciale, expédiée par la fonction OrderSend();
- MqlTradeTransaction contient la description de la transaction commerciale;
- MqlTick est destinée à la réception rapide de l'information la plus réclamée sur les prix en cours.

© 2000-2015, MetaQuotes Software Corp.

MqlDateTime

La structure de la date comprend huit champs du type int.

```
struct MqlDateTime
  {
   int year;            // année
   int mon;             // mois
   int day;             // jour
   int hour;            // heure
   int min;             // minutes
   int sec;             // secondes
   int day_of_week;     // jour de la semaine (0-dimanche, 1-lundi)
   int day_of_year;     // numéro d'ordre en année (le 1 février - c'est le 32-ième jour dans l'année
  };
```

Note

Le numéro d'ordre à l'année day_of_year à l'année bissextile, à partir de mars, se distinguera du numéro d'ordre du jour correspondant à l'année non -bissextile.

Exemple:

```
void OnStart()
  {
//---
   datetime date1=D'2008.03.01';
   datetime date2=D'2009.03.01';

   MqlDateTime str1,str2;
   TimeToStruct(date1,str1);
   TimeToStruct(date2,str2);
   printf("%02d.%02d.%4d, day of year = %d",str1.day,str1.mon,
          str1.year,str1.day_of_year);
   printf("%02d.%02d.%4d, day of year = %d",str2.day,str2.mon,
          str2.year,str2.day_of_year);
  }
/*  Résultat
   01.03.2008, day of year = 60
   01.03.2009, day of year = 59
*/
```

Voir aussi

TimeToStruct, Les structures et les classes

© 2000-2015, MetaQuotes Software Corp.

La structure des paramètres d'entrée de l'indicateur (MqlParam)

La structure MqlParam est spécialement élaborée pour la transmission des paramètres d'entrée à la création du handle de l'indicateur technique à l'aide de la fonction IndicatorCreate().

```
struct MqlParam
  {
   ENUM_DATATYPE     type;                  // type du paramètre d'entrée,la valeur de l'énumérati
   long              integer_value;         // champ pour le stockage de la valeur entière
   double            double_value;          // champ pour le stockage de la valeur double ou float
   string            string_value;          // champ pour le stockage de la valeur du type de lign
  };
```

Tous les paramètres d'entrée de l'indicateur sont transmis en forme du tableau comme MqlParam, le champ *type* de chaque élément de ce tableau indique le type de données, transmis par cet élément. Les valeurs d'indicateur doivent être d'abord placées dans les champs appropriés pour chaque élément (au *integer_value*, au *double_value* ou au *string_value*) selon quelle valeur d'énumération ENUM_DATATYPE est spécifiée dans le champ *type*.

Si la valeur IND_CUSTOM est transmise à la fonction IndicatorCreate() par le troisième paramètre comme le type de l'indicateur, le premier élément de tableau des paramètres d'entrée doit avoir le champ *type* avec la valeur TYPE_STRING de l'énumération ENUM_DATATYPE,et le champ *string_value* doit contenir le nom de l'indicateur de l'utilisateur.

© 2000-2015, MetaQuotes Software Corp.

MqlRates

La structure pour le stockage de l'information sur les prix, les volumes et le spread.

```
struct MqlRates
  {
   datetime time;         // temps du début de la période
   double   open;         // prix de l'ouverture
   double   high;         // le plus haut prix pour la période
   double   low;          // le plus petit prix pour la période
   double   close;        // prix de la clôture
   long     tick_volume;  // volume de tick
   int      spread;       // spread
   long     real_volume;  // volume de titres
  };
```

Exemple:

```
void OnStart()
  {
   MqlRates rates[];
   int copied=CopyRates(NULL,0,0,100,rates);
   if(copied<=0)
      Print("L'erreur du copiage des données de prix ",GetLastError());
   else Print("A copié ",ArraySize(rates)," des barres");
  }
```

Voir aussi

CopyRates, L'accès aux séries temporelles

© 2000-2015, MetaQuotes Software Corp.

MqlBookInfo

La structure accordant l'information dans le profondeur de marché.

```
struct MqlBookInfo
  {
   ENUM_BOOK_TYPE    type;        // type d'ordre de l'énumération ENUM_BOOK_TYPE
   double            price;       // prix
   long              volume;      // volume
  };
```

Note

La structure MqlBookInfo est prédéterminée, c'est pourquoi sa déclaration et la description ne sont pas demandées. Pour utiliser la structure, il suffit de déclarer la variable du type donné.

Le profondeur de marché est accessible non pour tous les instruments financiers.

Exemple:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo selon ",Symbol());
     }
   else
     {
      Print("On ne réussit pas à recevoir le contenu du profondeur de marché selon le
     }
```

Voir aussi

MarketBookAdd, MarketBookRelease, MarketBookGet, Les ordres commerciaux dans le profondeur de marché, Les types des données

© 2000-2015, MetaQuotes Software Corp.

La structure de la requête commerciale (MqlTradeRequest)

L'interaction du terminal de client et le serveur commercial pour la tenue des opérations du positionnement des ordres est produite au moyen des demandes commerciales. La demande est représentée par une structure spéciale prédéterminée MqlTradeRequest, qui contient tous les champs nécessaires pour la conclusion des actes de commerce. Le résultat du traitement de la demande est présenté par la structure MqlTradeResult.

```
struct MqlTradeRequest
  {
   ENUM_TRADE_REQUEST_ACTIONS    action;           // Type d'opération commercial
   ulong                         magic;            // Marque de l'expert (identificateur magic numbe
   ulong                         order;            // Ticket de l'ordre
   string                        symbol;           // Nom de l'instrument commercial
   double                        volume;           // Volume demandé du marché dans les lots
   double                        price;            // Prix
   double                        stoplimit;        // Niveau StopLimit de l'ordre
   double                        sl;               // Niveau Stop Loss de l'ordre
   double                        tp;               // Niveau Take Profit de l'ordre
   ulong                         deviation;        // Déviation possible maximal du prix demandé
   ENUM_ORDER_TYPE               type;             // Type de l'ordre
   ENUM_ORDER_TYPE_FILLING       type_filling;     // Type de l'ordre selon l'exécution
   ENUM_ORDER_TYPE_TIME          type_time;        // Type de l'ordre par temps de l'action
   datetime                      expiration;       // Délai de l'expiration de l'ordre(pour les ordr
   string                        comment;          // Commentaire sur l'ordre
  };
```

Description des champs

Champ	Description
action	Le type de la transaction commerciale. La valeur peut être une des valeurs de l'énumération ENUM_TRADE_REQUEST_ACTIONS
magic	L'identificateur de l'expert. Permet d'organiser le traitement analytique des ordres commerciaux. Chaque expert peut exposer l'identificateur personnel unique à l'expédition de la demande commerciale
order	Le ticket de l'ordre. Il est nécessaire à la modification des ordres remis
symbol	Le nom de l'instrument commercial, selon lequel l'ordre est présenté. Il n'est pas nécessaire aux opérations de la modification des ordres et la clôture des positions
volume	Le volume demandé du marché dans les lots. La signification réelle du volume à l'ouverture du marché dépendra du type de l'ordre selon l'exécution.
price	Le prix, à l'acquisition de lequel l'ordre doit être exécuté. Le prix, à l'acquisition de lequel l'ordre

© 2000-2015, MetaQuotes Software Corp.

	doit être exécuté. Il ne faut pas indiquer le prix pour les ordres de marché selon les instruments avec le type de l'exécution "Market Execution" (SYMBOL_TRADE_EXECUTION_MARKET), ayant le type TRADE_ACTION_DEAL
stoplimit	Le prix, selon lequel sera exposé l'ordre remis StopLimit, à l'acquisition au prix de la valeur price (cette condition est obligatoire). Jusqu'à ce moment l'ordre remis au système commercial n'est pas déduit
sl	Le prix, selon lequel fonctionnera l'ordre Stop Loss au mouvement du prix dans la direction défavorable
tp	Le prix, selon lequel fonctionnera l'ordre Take Profit au mouvement du prix dans la direction favorable
deviation	Le rejet maximum acceptable du prix demandé, donné dans les points
type	Le type d'ordre. Peut être une des valeurs d'énumération ENUM_ORDER_TYPE
type_filling	Le type d'exécution d'ordre. Peut être une de l'énumération des valeurs ENUM_ORDER_TYPE_FILLING
type_time	Le type d'expiration d'ordre. Peut être une de l'énumération des valeurs ENUM_ORDER_TYPE_TIME
expiration	Le délai de l'expiration de l'ordre remis (pour les ordres du type ORDER_TIME_SPECIFIED)
comment	Le commentaire pour l'ordre

Pour l'expédition des ordres sur l'accomplissement des transactions commerciales il est nécessaire d'utiliser la fonction OrderSend(). Pour chaque transaction commerciale il est nécessaire d'indiquer les champs obligatoires et on peut remplir les champs des options. Il est prévu au total sept variantes de l'expédition de la demande commerciale:

Request Execution

L'ordre commercial sur l'ouverture de la position en régime Request Execution (le régime du commerce à la demande des valeurs présentes). On demande l'indication de 9 champs:

- action
- symbol
- volume
- price
- sl
- tp

© 2000-2015, MetaQuotes Software Corp.

- deviation
- type
- type_filling

On peut aussi spécifier les valeurs des champs magic et comment.

Instant Execution

L'ordre commercial sur l'ouverture de la position en régime Instant Execution (le commerce par les prix d'écoulement). On demande l'indication de 9 champs:

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type_filling

On peut aussi spécifier les valeurs des champs magic et comment.

Market Execution

L'ordre commercial sur l'ouverture de la position en mode Market Execution (le mode de l'exécution des ordres commerciaux selon le marché).On demande l'indication de 5 champs:

- action
- symbol
- volume
- type
- type_filling

On peut aussi spécifier les valeurs des champs magic et comment.

Exchange Execution

L'ordre commercial sur l'ouverture de la position en mode Exchange Execution (le mode boursier de l'exécution des ordres commerciaux). On demande l'indication de 5 champs:

- action
- symbol
- volume
- type
- type_filling

On peut aussi spécifier les valeurs des champs magic et comment.

SL & TP Modification

L'ordre commercial sur la modification des niveaux StopLoss et/ou TakeProfit. On demande l'indication de 4 champs:

- action
- symbol
- sl

© 2000-2015, MetaQuotes Software Corp.

- tp

Pending Order

L'ordre commercial sur l'installation de l'ordre remis. On demande l'indication de 11 champs::

- action
- symbol
- volume
- price
- stoplimit
- sl
- tp
- type
- type_filling
- type_time
- expiration

On peut aussi spécifier les valeurs des champs magic et comment.

Modify Pending Order

L'ordre commercial sur la modification des niveaux des prix de l'ordre remis. On demande l'indication de 7 champs:

- action
- order
- price
- sl
- tp
- type_time
- expiration

Delete Pending Order

L'ordre commercial de supprimer l'ordre remis. On demande l'indication de 2 champs:

- action
- order

Voir aussi

Les structures et les classes, Les fonctions commerciales, Les propriété des ordres

© 2000-2015, MetaQuotes Software Corp.

La structure des résultats du contrôle de la demande commerciale(MqlTradeCheckResult)

Avant d'envoyerau serveur commercial la demande sur l'opération commerciale,, il est recommandé de la vérifier. Le contrôle se réalise par la fonction OrderCheck(), à laquelle on transmet la demande contrôlée et la variable du type de la structure MqlTradeCheckResult. A cette variable on inscrira le résultat du contrôle. /t8>

```
struct MqlTradeCheckResult
  {
   uint         retcode;             // Le code de la réponse
   double       balance;             // La balance après l'exécution de l'opération
   double       equity;              // Les capitaux propres après l'exécution de l'opération
   double       profit;              // Le bénéfice nageant
   double       margin;              // Les exigences de la marge
   double       margin_free;         // La marge libre
   double       margin_level;        // Le niveau de la marge
   string       comment;             // Le commentaire sur le code de la réponse (la description de
  };
```

La description des champs

Le champ	La description
retcode	Le code du retour
balance	La valeur de la balance, qui sera après l'exécution de l'opération commerciale
equity	La valeur des moyens propres, qui sera après l'exécution de l'opération commerciale
profit	La valeur du bénéfice nageant, qui sera après l'exécution de l'opération commerciale
margin	Le montant de la marge nécessaire à l'opération commerciale demandée
margin_free	Le montant des moyens libres propres, lesquels resteront après l'exécution de l'opération commerciale demandée
margin_level	Le niveau de la marge, qui s'établira après l'exécution de l'opération commerciale demandée
comment	Le commentaire sur le code de la réponse, la description de l'erreur

Voir aussi

La structure de la demande commerciale, La structure pour la réception des prix présents, OrderSend, OrderCheck

© 2000-2015, MetaQuotes Software Corp.

La structure du résultat de la requête commerciale (MqlTradeResult)

En réponse à la requête commerciale du positionnement de l'ordre au système commercial, le serveur commercial rend les données contenant l'information sur le résultat du traitement de la requête commerciale en forme de la structure spéciale prédéterminée MqlTradeResult.

```
struct MqlTradeResult
  {
   uint     retcode;          // Code du résultat de l'opération
   ulong    deal;             // Ticket du marché, s'il est exécuté
   ulong    order;            // Ticket de l'ordre, s'il est exposé
   double   volume;           // Volume du marché confirmé par le broker
   double   price;            // Prix dans le marché, confirmé par le broker
   double   bid;              // Prix courant en cours de l'offre (prix de recours)
   double   ask;              // Prix courant en cours de la requête (prix de recours)
   string   comment;          // Commentaire du broker pour l'opération (par défaut il est rempli pa
   uint     request_id;       // l'identificateur de la demande, s'établit par le terminal à l'envoi
  };
```

Description des champs

Champ	Description
retcode	Le code du retour du serveur commercial
deal	Le ticket du marché, s'il est faite. Il informe de la transaction commerciale TRADE_ACTION_DEAL
order	Le ticket du marché, s'il est exposé. Il informe de la transaction commerciale TRADE_ACTION_PENDING
volume	Le volume du marché confirmé par le broker. Dépend du type de l'ordre selon l'exécution
price	Le prix dans le marché, confirmé par le broker. Dépend du champ *deviation* dans la requête commerciale et/ou du type de la transaction commerciale
bid	Le prix courant en cours de l'offre (les prix de la recitation)
ask	Le prix courant en cours de la requête (les prix de la recitation)
comment	Le commentaire du broker pour l'opération (par défaut il est rempli par la description le code du retour du serveur commercial)
request_id	L'identificateur de la demande mis par le terminal à l'envoi sur le serveur commercial

Le résultat de la transaction commerciale revient à la variable comme MqlTradeResult, qui est

© 2000-2015, MetaQuotes Software Corp.

transmise par le deuxième paramètre à la fonction OrderSend() pour la tenue des transactions commerciales.

Le terminal inscrit l'identificateurde la demande dans le champ request_id à son envoi sur le serveur commercial par les fonctions OrdersSend() et OrderSendAsync(). Du serveur commercial le terminal reçoit les messages sur les transactions commerciales faites et les transmet au traitement dans la fonction OnTradeTransaction(), qui contient comme paramètres:

- la description de la transaction commerciale MqlTradeTransaction;
- la description de la demande commerciale , envoyé par la fonction OrderSend() ou OrdersSendAsync(). L'identificateur de la demande est envoyé par le terminal sur le serveur commercial, et la demande elle-même et son request_id se sauvegardent dans la mémoire du terminal;
- Le résultat de l'exécution de la demande commerciale en forme de la structure MqlTradeResult, où le champ request_id contient l'identificateur de cette demande.

La fonction OnTradeTransaction() reçoit trois paramètres d'entrée, mais il faut analyser les derniers deux paramètres seulement pour les transactions commerciales, ayant le type TRADE_TRANSACTION_REQUEST. Dans tous les autres cas les données sur la demande commerciale et le résultat de son exécution ne sont pas remplies. L'exemple de l'analyse des paramètres est amené dans le paragraphe La structure de la transaction commerciale.

L'installation de l'identificateur request_id par le terminal pour la demande commerciale à son envoi sur le serveur est destinée en premier lieu au travail avec la fonction asynchrone OrderSendAsync(). Cet identificateur permet de lier l'action éxécutée (l'appel des fonctions OrderSend ou OrderSendAsync) à la suite de cette action transmise dansOnTradeTransaction().

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Expédition de la requête commerciale avec le traitement du résultat |
//+------------------------------------------------------------------+
bool MyOrderSend(MqlTradeRequest request,MqlTradeResult result)
  {
//--- enlevons le code de la dernière erreur au zéro
   ResetLastError();
//--- expédierons la requête
   bool success=OrderSend(request,result);
//--- si le résultat est mauvais  - essayons d'apprendre pourquoi
   if(!success)
     {
      int answer=result.retcode;
      Print("TradeLog:Trade request failed. Error = ",GetLastError());
      switch(answer)
        {
         //--- recitation
         case 10004:
           {
            Print("TRADE_RETCODE_REQUOTE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid =",result.bid);
            break;
           }
         //--- ordre n'est pas accepté par le serveur
         case 10006:
           {
            Print("TRADE_RETCODE_REJECT");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //---  prix incorrect
         case 10015:
           {
            Print("TRADE_RETCODE_INVALID_PRICE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid =",result.bid);
            break;
           }
         //--- incorrect SL et/ou TP
         case 10016:
           {
            Print("TRADE_RETCODE_INVALID_STOPS");
            Print("request.sl = ",request.sl," request.tp = ",request.tp);
            Print("result.ask = ",result.ask," result.bid = ",result.bid);
            break;
           }
         //--- volume incorrect
         case 10014:
           {
            Print("TRADE_RETCODE_INVALID_VOLUME");
            Print("request.volume = ",request.volume,"   result.volume = ",
                  result.volume);
            break;
           }
         //--- il ne suffit pas l'argent sur la transaction commerciale
         case 10019:
           {
            Print("TRADE_RETCODE_NO_MONEY");
            Print("request.volume = ",request.volume,"   result.volume = ",
```

© 2000-2015, MetaQuotes Software Corp.

```
                  result.volume,"   result.comment = ",result.comment);
            break;
           }
         //--- autre raison, informons du code de la réponse du serveur

         default:
           {
            Print("Other answer = ",answer);
           }
        }
      //--- informons du mauvais résultat de la requête commerciale par le retour fals
      return(false);
     }
//--- OrderSend() a rendu true - répéterons la réponse
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

La structure de la transaction commerciale (MqlTradeTransaction)

A la suite de l'exécution des actions définies avec le compte commercial, son état change. Ces actions comprennent:

- L'envoi de la demande commerciale par n'importe quelle application MQL5- dans le terminal de client à l'aide des fonctions OrderSend et OrderSendAsync et son exécution ultérieure;
- L'envoi de la demande commerciale par l'interface graphique du terminal et son exécution ultérieure;
- Le fonctionnement des ordres remis et des ordres -stop sur le serveur;
- L'exécution des opérations sur le côté du serveur commercial.

A la suite de ces actions, les transactions commerciales sont exécutées pour le compte:

- le traitement de la demande commerciale;
- le changement des ordres ouverts;
- le changement de l'histoire des ordres;
- le changement de l'histoire des marchés;
- le changement des positions.

Par exemple, à l'envoi de l'ordre de marché sur l'achat, il est traité, l'ordre correspondant sur l'achat est créé pour le compte, se passe l'exécution de l'ordre, son suppression de la liste des ouverts, l'ajout à l'histoire des ordres, ensuite le marché correspondant est ajouté à l'histoire et une nouvelle position est créé. Toutes ces actions sont les transactions commerciales.

Pour la réception des transactions commerciales, appliquées au compte, dans MQL5 il y a un gestionnaire spécial OnTradeTransaction(). La structure MqlTradeTransaction décrivant les transactions commerciales est transmise au premier paramètre de ce gestionnaire.

```
struct MqlTradeTransaction
  {
   ulong                     deal;             // Le ticket du marché
   ulong                     order;            // Le ticket de l'ordre
   string                    symbol;           // Le nom de l'instrument commercial
   ENUM_TRADE_TRANSACTION_TYPE type;           // Le type de la transaction commerciale
   ENUM_ORDER_TYPE           order_type;       // Le type de l'ordre
   ENUM_ORDER_STATE          order_state;      // L'état de l'ordre
   ENUM_DEAL_TYPE            deal_type;        // Le type du marché
   ENUM_ORDER_TYPE_TIME      time_type;        // Le type de l'ordre selon le temps de l'action
   datetime                  time_expiration;  // La date de l'expiration de l'ordre
   double                    price;            // Le prix
   double                    price_trigger;    // Le prix du fonctionnement de l'ordre stop limi
   double                    price_sl;         // Le niveau Stop Loss
   double                    price_tp;         // Le niveau Take Profit
   double                    volume;           // Le volume en lots
  };
```

La description des champs

Le champ	La description
deal	Le ticket du marché.

© 2000-2015, MetaQuotes Software Corp.

order	Le ticket de l'ordre.
symbol	Le nom de l'instrument commercial, selon lequel la transaction est faite.
type	Le type de la transaction commerciale. La valeur peut être une des valeurs de l'énumération ENUM_TRADE_TRANSACTION_TYPE.
order_type	Le type de l'ordre commercial. La valeur peut être une des valeurs de l'énumération ENUM_ORDER_TYPE.
order_state	L'état de l'ordre commercial. La valeur peut être une des valeurs de l'énumération ENUM_ORDER_STATE.
deal_type	Le type du marché. La valeur peut être une des valeurs de l'énumération ENUM_DEAL_TYPE.
type_time	Le type de l'ordre selon l'expiration. La valeur peut être une des valeurs de l'énumération ENUM_ORDER_TYPE_TIME.
time_expiration	La date de l'expiration de l'ordre remis (pour les ordres du typeORDER_TIME_SPECIFIED et ORDER_TIME_SPECIFIED_DAY).
price	Le prix. En fonction du type de la transaction commerciale il peut être le prix de l'ordre, du marché ou de la position.
price_trigger	Le prix-stop (le prix du fonctionnement) de l'ordre stop limit (ORDER_TYPE_BUY_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT).
sl	Le prix Stop Loss. En fonction du type de la transaction commerciale il peut se rapporter à l'ordre, au marché ou à la position.
tp	Le prix Take Profit. En fonction du type de la transaction commerciale il peut se rapporter à l'ordre, au marché ou à la position.
volume	Le volume en lots. En fonction du type de la transaction commerciale il peut indiquer au volume courant de l'ordre, le volume du marché ou le volume de la position.

Le paramètre essentiel pour l'analyse de transaction reçue est son type spécifié au champ**type**. Par exemple, si la transaction c'est le type TRADE_TRANSACTION_REQUEST (le résultat du traitement de la demande commerciale par le serveur est reçu), la structure a seulement le champ rempli **type**, il ne faut pas analyser les autres champs. Dans ce cas on peut faire l'analyse de deux paramètres supplémentaires **request** et **result**, qui sont transmis dans le gestionnaire OnTradeTransaction(),

© 2000-2015, MetaQuotes Software Corp.

comme c'est montré dans l'exemple plus bas.

En connaissant le type de l'opération commerciale, on peut décision de l'analyse de l'état courant des ordres, des positions et des marchés sur le compte commercial. Il est nécessaire d'avoir en vue qu'une demande commerciale, envoyé au serveur du terminal peut engendrer quelques transactions commerciales, dont l'ordre successif de l'entrée au terminal n'est pas garanti.

La structure MqlTradeTransaction est remplie différemment en fonction du type de la transaction commerciale (ENUM_TRADE_TRANSACTION_TYPE):

TRADE_TRANSACTION_ORDER_* et TRADE_TRANSACTION_HISTORY_*

Les champs suivants sont remplis dans la structure MqlTradeTransaction pour les transactions commerciales, concernant le traitement des ordres ouverts (TRADE_TRANSACTION_ORDER_ADD, TRADE_TRANSACTION_ORDER_UPDATE et TRADE_TRANSACTION_ORDER_DELETE) et l'histoire des ordres (TRADE_TRANSACTION_HISTORY_ADD, TRADE_TRANSACTION_HISTORY_UPDATE, TRADE_TRANSACTION_HISTORY_DELETE):

- order - le ticket de l'ordre;
- le symbol - le nom de l'instrument financier dans l'ordre;
- type - le type de la transaction commerciale;
- order_type - le type de l'ordre;
- orders_state - l'état actuel de l'ordre;
- time_type - le type de l'expiration de l'ordre;
- time_expiration - le temps de l'expiration de l'ordre (pour les ordres avec le type de l'expiration ORDER_TIME_SPECIFIED et ORDER_TIME_SPECIFIED_DAY);
- price - le prix dans l'ordre, indiqué par le client;
- price_trigger - le prix-stop du fonctionnement de l'ordre-stop limite (seulement pour ORDER_TYPE_BUY_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT);
- price_sl - le prix Stop Loss de l'ordre (est rempli, s'il est indiquée dans l'ordre);
- price_tp - le prix Take Profit de l'ordre (est rempli, s'il est indiquée dans l'ordre);
- volume - le volume courant de l'ordre (non exécuté). On peut savoir le volume initial de l'ordre de l'histoire des ordres à l'aide des fonctionsHistoryOrders*.

TRADE_TRANSACTION_DEAL_*

Les champs suivants sont remplis dans la structure MqlTradeTransaction pour les transactions commerciales, concernant le traitement des marchés (TRADE_TRANSACTION_DEAL_ADD, TRADE_TRANSACTION_DEAL_UPDATE et TRADE_TRANSACTION_DEAL_DELETE):

- deal - le ticket du marché;
- order - le ticket de l'ordre, à la base de lequel le marché a été exécuté;
- le symbol - le nom de l'instrument financier dans l'ordre;
- type - le type de la transaction commerciale;
- deal_type - le type du marché;
- price - le prix, selon lequel le marché a été exécuté;
- price_sl - le prix Stop Loss (est rempli, s'il est indiquée dans l'ordre, à la base de lequel le marché a été exécuté);
- price_tp - le prix Take Profit (est rempli, s'il est indiquée dans l'ordre, à la base de lequel le marché a été exécuté);
- volume - le volume du marché en lots.

© 2000-2015, MetaQuotes Software Corp.

TRADE_TRANSACTION_POSITION

Les champs suivants sont remplis dans la structure MqlTradeTransaction pour les transactions commerciales, concernant les changements des positions, qui ne sont pas liés à l'exécution des marchés (TRADE_TRANSACTION_POSITION):

- symbol - le nom de l'instrument de la position;
- type - le type de la transaction commerciale;
- deal_type - le type de la position (DEAL_TYPE_BUY ou DEAL_TYPE_SELL);
- price - le prix moyen pondéré de l'ouverture de la position;
- price_sl - le prix Stop Loss;
- price_tp - le prix Take Profit;
- volume - le volume de la position en lots, s'il était changé.

> Le changement de la position (l'ajout, le changement ou la liquidation) à la suite de l'exécution du marché n'entraîne pas l'apparition de la transaction TRADE_TRANSACTION_POSITION.

TRADE_TRANSACTION_REQUEST

Seulement un champ est rempli dans la structure MqlTradeTransaction pour les transactions commerciales, décrivant le fait que la demande commerciale est tratée par le serveur, et le résultat de son traitement est reçu (TRADE_TRANSACTION_REQUEST):

- type - le type de la transaction commerciale;

> Pour les transaction de ce type il est nécessaire d'analyser seulement un champ - type (le type de la transaction commerciale). Pour la réception de l'information supplémentaire il est nécessaire d'analyser les deuxièmes et troisièmes paramètres de la fonction OnTradeTransaction (request et result).

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
input int MagicNumber=1234567;

//--- connectons la classe commerciale CTrade et annonçons la variable de ce type
#include <Trade\Trade.mqh>
CTrade trade;
//--- les drapeaux pour l'installation et lq suppression de l'ordre remis
bool pending_done=false;
bool pending_deleted=false;
//---sauvegarons ici le ticket de l'ordre remis
ulong order_ticket;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- établissons MagicNumber, par lequel tous nos ordres serons marqués
   trade.SetExpertMagicNumber(MagicNumber);
//--- expédierons les demandes commerciales en mode asynchrone à l'aide de la fonctior
   trade.SetAsyncMode(true);
//---initialisons la variable par zéro
   order_ticket=0;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---l'installation de l'ordre remis
   if(!pending_done)
     {
      double ask=SymbolInfoDouble(_Symbol,SYMBOL_ASK);
      double buy_stop_price=NormalizeDouble(ask+1000*_Point,(int)SymbolInfoInteger(_Sy
      bool res=trade.BuyStop(0.1,buy_stop_price,_Symbol);
      //--- si la fonction BuyStop() a travaillé avec succès
      if(res)
        {
         pending_done=true;
         //---recevrons le résultat de l'envoi de la demande à partir de ctrade
         MqlTradeResult trade_result;
         trade.Result(trade_result);
         //---recevrons request_id pour la demande envoyé
         uint request_id=trade_result.request_id;
         Print("La demande pour l'installation de l'ordre remis a été envoyé. L'identi
         //--- retiendrons le ticket de l'ordre (à l'utilisation du mode asynchrone de
         order_ticket=trade_result.order;
         //--- Tout est fait, c'est pourquoi nous sortons du gestionnaire OnTick() ava
         return;
        }
     }
//---la suppression de l'ordre remis
   if(!pending_deleted)
      //--- la vérification supplémentaire
      if(pending_done && (order_ticket!=0))
        {
         //---tenterons du supprimer l'ordre remis
         bool res=trade.OrderDelete(order_ticket);
         Print("OrderDelete=",res);
         //--- à l'envoi réussi de la demande pour la suppression
         if(res)
```

© 2000-2015, MetaQuotes Software Corp.

```
            {
             pending_deleted=true;
             //--- recevrons le résultat de l'exécution de la demande
             MqlTradeResult trade_result;
             trade.Result(trade_result);
             //--- sortirons l'identificateur de la demande du résultat
             uint request_id=trade_result.request_id;
             //--- déduisons au Journal
             Print("La demande pour la suppression de l'ordre remis a été envoyé #",orc
                   ". L'identificateur de la demande Request_ID=",request_id,
                   "\r\n");
             //--- inscrirons le ticket de l'ordre du résultat de la demande
             order_ticket=trade_result.order;
            }
         }
//---
  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- recevrons le type de la transaction en forme de la valeur de l'énumération
   ENUM_TRADE_TRANSACTION_TYPE type=(ENUM_TRADE_TRANSACTION_TYPE)trans.type;
//--- si la transaction c'est le résultat du traitement de la demande, nous déduisons
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      Print(EnumToString(type));
      //--- déduisons la description de chaîne de la demande traitée
      Print("------------RequestDescription\r\n",RequestDescription(request));
      //--- déduisons la description du résultat de la demande
      Print("------------ResultDescription\r\n",TradeResultDescription(result));
      //--- retiendrons le ticket de l'ordre pour son suppression au traitement suivar
      if(result.order!=0)
        {
         //---supprimons cet ordre selon son ticket à l'appel suivant OnTick()
         order_ticket=result.order;
         Print(" Le ticket de l'ordre remis ",order_ticket,"\r\n");
        }
     }
   else // déduisons la description complète pour les transactions d'un autre type
//--- déduisons la description de la transaction reçue au Journal
      Print("------------TransactionDescription\r\n",TransactionDescription(trans));

//---
  }
//+------------------------------------------------------------------+
//| Rend une description textuelle de la transaction                 |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans)
  {
//---
   string desc=EnumToString(trans.type)+"\r\n";
   desc+="Symbol: "+trans.symbol+"\r\n";
   desc+="Deal ticket: "+(string)trans.deal+"\r\n";
   desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
   desc+="Order ticket: "+(string)trans.order+"\r\n";
   desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
   desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
```

© 2000-2015, MetaQuotes Software Corp.

```
   desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
   desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
   desc+="Price trigger: "+StringFormat("%G",trans.price_trigger)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
   desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
//--- rendons la chaîne réçue
   return desc;
  }
//+------------------------------------------------------------------+
//| Rend la description textuelle de la demande commerciale          |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request)
  {
//---
   string desc=EnumToString(request.action)+"\r\n";
   desc+="Symbol: "+request.symbol+"\r\n";
   desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
   desc+="Order ticket: "+(string)request.order+"\r\n";
   desc+="Order type: "+EnumToString(request.type)+"\r\n";
   desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
   desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
   desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
   desc+="Deviation points: "+StringFormat("%G",request.deviation)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
   desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
   desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
   desc+="Comment: "+request.comment+"\r\n";
//--- rendons la chaîne réçue
   return desc;
  }
//+------------------------------------------------------------------+
//| Rend la description textuelle du résultat du traitement de la    |
//| demande                                                          |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result)
  {
//---
   string desc="Retcode "+(string)result.retcode+"\r\n";
   desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
   desc+="Order ticket: "+(string)result.order+"\r\n";
   desc+="Deal ticket: "+(string)result.deal+"\r\n";
   desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
   desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
   desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
   desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
   desc+="Comment: "+result.comment+"\r\n";
//--- rendons la chaîne réçue
   return desc;
  }
```

Voir aussi

Les types des transactions commerciales, OnTradeTransaction()

© 2000-2015, MetaQuotes Software Corp.

La structure pour la réception des valeurs présentes (MqlTick)

La structure pour la conservation des derniers prix selon le symbole. Elle est destinée à la réception rapide de l'information la plus réclamée sur les valeurs présentes.

```
struct MqlTick
  {
   datetime     time;          // Temps de la dernière mise à jour des prix
   double       bid;           // Valeur présente Bid
   double       ask;           // Valeur présente Ask
   double       last;          // Valeur présente du dernier marché (Last)
   ulong        volume;        // Volume pour la valeur présente Last
   long         time_msc;      // Time of a price last update in milliseconds
   uint         flag           // Tick flags
  };
```

La variable du type MqlTick permet pour un appel de la fonction SymbolInfoTick() recevoir les valeurs Ask, Bid, Last et Volume.

The parameters of each tick are filled in regardless of whether there are changes compared to the previous tick. Thus, it is possible to find out a correct price for any moment in the past without the need to search for previous values at the tick history. For example, even if only a Bid price changes during a tick arrival, the structure still contains other parameters as well, including the previous Ask price, volume, etc.

You can analyze the tick flags to find out what data have been changed exactly:

- TICK_FLAG_BID - tick has changed a Bid price
- TICK_FLAG_ASK - a tick has changed an Ask price
- TICK_FLAG_LAST - a tick has changed the last deal price
- TICK_FLAG_VOLUME - a tick has changed a volume
- TICK_FLAG_BUY - a tick is a result of a buy deal
- TICK_FLAG_SELL - a tick is a result of a sell deal

Exemple:

```
void OnTick()
  {
   MqlTick last_tick;
//---
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      Print(last_tick.time,": Bid = ",last_tick.bid,
            " Ask = ",last_tick.ask,"  Volume = ",last_tick.volume);
     }
   else Print("SymbolInfoTick() failed, error = ",GetLastError());
//---
  }
```

Voir aussi

© 2000-2015, MetaQuotes Software Corp.

Les structures et les classes, CopyTicks(), SymbolInfoTick()

© 2000-2015, MetaQuotes Software Corp.

Les codes des erreurs et des avertissements

Le paragraphe contient les descriptions suivantes:

- Les codes du retour du serveur commercial - l'analyse des résultats de l'expédition de la requête commerciale, expédiée par la fonction OrderSend();
- Les avertissements de compilateur- les codes des messages préventifs déduits à la compilation (ne sont pas les erreurs);
- Les erreurs de la compilation - les codes de messages d'erreur à une tentative infructueuse de la compilation;
- Les erreurs du temps de l'exécution - les codes des erreurs à l'exécution du programme mql5, lesquels on peut recevoir à l'aide de la fonction GetLastErtror().

© 2000-2015, MetaQuotes Software Corp.

Les codes du retour du serveur commercial

Tous les ordres sur l'accomplissement des transactions commerciales partent en forme de la structure de la demande commerciale MqlTradeRequest à l'aide de la fonction OrderSend(). Le résultat de l'exécution de cette fonction se place à la structure MqlTradeResult, le champ *retcode* qui contient le code du retour du serveur commercial.

Code	Identificateur	Description
10004	TRADE_RETCODE_REQUOTE	La recitation
10006	TRADE_RETCODE_REJECT	La requête rejetée
10007	TRADE_RETCODE_CANCEL	La requête est annulé par un broker
10008	TRADE_RETCODE_PLACED	L'ordre a placé
10009	TRADE_RETCODE_DONE	La requête est accomplie
10010	TRADE_RETCODE_DONE_PARTIAL	La requête est partiellement accomplie
10011	TRADE_RETCODE_ERROR	L'erreur de traitement de requête
10012	TRADE_RETCODE_TIMEOUT	La requête est supprimée à l'expiration du temps
10013	TRADE_RETCODE_INVALID	La requête incorrecte
10014	TRADE_RETCODE_INVALID_VOLUME	Le volume incorrect dans la requête
10015	TRADE_RETCODE_INVALID_PRICE	Le prix incorrect dans la requête
10016	TRADE_RETCODE_INVALID_STOPS	Les Stops incorrects dans la requête
10017	TRADE_RETCODE_TRADE_DISABLED	Le commerce est interdit
10018	TRADE_RETCODE_MARKET_CLOSED	Le marché est fermé
10019	TRADE_RETCODE_NO_MONEY	Il n'y a pas de ressources suffisantes pour l'exécution de la requête
10020	TRADE_RETCODE_PRICE_CHANGED	Les prix ont changé
10021	TRADE_RETCODE_PRICE_OFF	Les cotations pour le traitement de la requête manquent
10022	TRADE_RETCODE_INVALID_EXPIRATION	La date incorrecte de l'expiration de l'ordre dans la

© 2000-2015, MetaQuotes Software Corp.

		requête
10023	TRADE_RETCODE_ORDER_CHANGED	L'état de l'ordre a changé
10024	TRADE_RETCODE_TOO_MANY_REQUESTS	Les requêtes trop fréquentes
10025	TRADE_RETCODE_NO_CHANGES	Il n'y a pas de changements dans la requête
10026	TRADE_RETCODE_SERVER_DISABLES_AT	L'autotrading est interdit par le serveur
10027	TRADE_RETCODE_CLIENT_DISABLES_AT	L'autotrading est interdit par le terminal de client
10028	TRADE_RETCODE_LOCKED	La requête est bloquée pour le traitement
10029	TRADE_RETCODE_FROZEN	L'ordre ou la position se sont gelés
10030	TRADE_RETCODE_INVALID_FILL	On indique le type non soutenu de l'exécution de l'ordre selon le reste
10031	TRADE_RETCODE_CONNECTION	Il n'y a pas de connection avec le serveur commercial.
10032	TRADE_RETCODE_ONLY_REAL	L'opération est permise seulement pour les comptes réels
10033	TRADE_RETCODE_LIMIT_ORDERS	La limite sur la quantité d'ordres remis est atteint
10034	TRADE_RETCODE_LIMIT_VOLUME	La limite sur le volume des ordres et les positions pour le symbole donné est atteint
10035	TRADE_RETCODE_INVALID_ORDER	le type de l'ordreincorrecte ou interdit
10036	TRADE_RETCODE_POSITION_CLOSED	La position avecPOSITION_IDENTIFIERindiquée est déjà fermée

© 2000-2015, MetaQuotes Software Corp.

Les avertissements de compilateur

Les avertissements de compilateur ont le caractère d'information et ne sont pas les messages d'erreur.

Numéro	Description
21	L'inscription incomplète de la date dans la ligne datetime
22	Les nombres faux dans la ligne datetime pour la date, les exigences: année 1970<=X<=3000 mois 0<X<=12 jour 0<X<= 31/30/28(29)....
23	Les nombres faux dans la ligne datetime pour le temps, les exigences: heure 0<=X<24 minute 0<=X<60
24	La couleur incorrecte dans le format RGB: un des composants RGB est moins que 0 ou plus que 255
25	Le symbole inconnu de l'héritage escape. Connus: \n \r \t \\ \" \' \X \x
26	Le volume est trop grand des variables locales (>512 Kb) les fonctions, diminuez leur quantité
29	L'énumération est déjà défini (le doublage) - les membres seront ajoutés à la première définition
30	La redéfinition de la macro
31	La variable est déclarée, mais n'est pas utilisée nulle part
32	Le constructeur doit avoir le type void
33	Le décomposeur doit avoir le type void
34	La constante n'entre pas au domaine des entiers (X>_UI64_MAX || X<_I64_MIN) et sera transcrite au type double
35	HEX est trop longue, plus que 16 symboles signifiants (se coupent les demi-bytes principaux)
36	Il n'y a pas d'aucun demi-byte à HEX dans la ligne "0x"
37	Il n'y a pas d'aucune fonction - on n'aura rien à accomplir
38	La variable non initialisée est utilisée

© 2000-2015, MetaQuotes Software Corp.

41	La fonction n'a pas le corps, mais elle n'est pas appelée
43	Les pertes des données à la transformation du type sont possibles. L'exemple: int x=(double)z;
44	La perte de l'exactitude (des données) à la transformation de la constante. L'exemple: int x=M_PI
45	La non-coïncidence des signes des opérandes dans les opérations de la comparaison. L'exemple: (char)c1>(uchar)c2
46	Les problèmes avec l'importation des fonctions - on demande la déclaration #import ou l'importation des fonctions est déjà fermée
47	La description trop grand - les symboles superflus ne seront pas insérés dans le fichier exécuté
48	Le nombre de tampons d'indicateur déclarés est moins qu'exigé
49	On n'indique pas la couleur pour le dessin de la série graphique dans l'indicateur
50	Il n'y a pas d'aucune série graphique pour l'affichage de l'indicateur
51	La fonction-gestionnaire 'OnStart" n'est pas découverte dans le script
52	La fonction-gestionnaire 'OnStart" est définie avec les paramètres incorrectes
53	La fonction 'OnStart' peut être défini seulement dans le script
54	La fonction 'OnInit' est définie avec les paramètres incorrectes
55	La fonction 'OnInit' n'est pas utilisé dans le script
56	La fonction 'OnDeinit' est définie avec les paramètres incorrectes
57	La fonction 'OnDeinit' n'est pas utilisé dans le script
58	Les deux fonctions 'OnCalculate' ont été définit. Sera utilisé OnCalculate() dans un tableau de prix
59	On a trouvé le débordement au calcul de la constante complexe entière

© 2000-2015, MetaQuotes Software Corp.

60	Probablement, la variable n'est pas initialisée.
61	Cette déclaration fait l'appel inaccessible à la variable locale, déclarée sur la chaîne indiquée
62	Cette déclaration fait l'appel inaccessible à la variable globale, déclarée sur la chaîne indiquée
63	Ne peut pas être utilisé pour les tableaux statiques
64	Cette déclaration fait l'appel inaccessible à la variable prédéterminé
65	La signification de l'expression est toujours true/false
66	L'utilisation de la variable ou de l'expression du type bool dans les opérations mathématiques est dangereuse
67	Le résultat de l'application de l'opérateur du moins unaire au type sans signeulong n'est pas défini
68	La version indiquée dans la propriété #property version, n'est pas autorisée à être placée dans le paragraphe Marché, le format juste #property version "XXX.YYY"
69	L'expression pour l'exécution selon la condition manque
70	Le type incorrecte rendu de la fonction ou les paramètres incorrects à la déclarationde la fonction-gestionnaire de l'événement
71	La conformation évidentvers un typeest nécessaire
72	Cette déclaration faite inaccessible l'accès direct au membre de la classe, annoncé sur la ligne indiquée. L'accès sera possible seulement à l'aide del'opération de la permission du contexte ::
73	La constante dans l'enregistrement binaire est trop grande, les catégories principales seront rejetées
74	Le paramètre dans la méthodede la classe héritée se distingue par le modificateur const, la fonction affiliée a surchargé la fonction du parent
75	La signification négative ou trop grande du déplacementde l'opération de bits du décalage,

© 2000-2015, MetaQuotes Software Corp.

	le résultat de l'exécution n'est pas défini
76	Function must return a value
77	void function returns a value
78	Not all control paths return a value
79	Expressions are not allowed on a global scale
80	Check operator_precedence for possible error; use parentheses to clarify precedence
81	Two OnCalCulate() are defined. OHLC version will be used
82	Struct has no members, size assigned to 1 byte
83	Return value of the function should be checked
84	Resource indicator is compiled for debugging. That slows down the performance. Please recompile the indicator to increase performance
85	Too great character code in the string, must be in the range 0 to 65535
86	Unrecognized character in the string
87	No indicator window property (setting the display in the main window or a subwindow) is defined. Property #property indicator_chart_window is applied

© 2000-2015, MetaQuotes Software Corp.

Les erreurs de la compilation

MetaEdtior 5, le rédacteur des programmes mql5 donne les messages d'erreur du programme, découverts par le compilateur inséré pendant la compilation. La liste de ces erreurs est donnée dans la table ci-dessous. Pour la compilation du code initial à exécuter appuyez **F7**. Les programmes qui contiennent des erreurs ne peuvent pas être compilés jusqu'à ce que les erreurs identifiées par le compilateur ne soient éliminées.

Numéro	Description
100	L'erreur de la lecture du fichier
101	L'erreur de l'ouverture *.EX5 du fichier sur l'inscription pour la sauvegarde
103	Pas assez de mémoire libre pour accomplir la compilation
104	L'unité syntaxique vide méconnue par le compilateur
105	Le nom incorrect du fichier dans #include
106	L'erreur de l'accès au fichier dans #include (probablement le fichier n'existe pas)
108	Le nom inconvenant pour #define
109	L'ordre inconnu du préprocesseur (sont accessibles #include,#define,#property,#import)
110	Le symbole inconnu pour compilateur
111	La fonction n'est pas réalisée (il y a la description, le corps est absent)
112	Les guillemets doubles sont manquées (")
113	La première équerre (<) ou les guillements doubles (") sont omises
114	Le guillemet (') est omis
115	L'équerre finale ">" est omise
116	On n'indique pas le type dans la déclaration
117	Il n'y a pas d'opérateur du retour "return" ou il se trouve non dans toutes les branches de l'exécution
118	On attendait la parenthèse ouvrante des paramètres de l'appel
119	L'erreur de l'inscription EX5
120	L'accès incorrect à l'élément du tableau

© 2000-2015, MetaQuotes Software Corp.

121	La fonction n'a pas le type "void" et l'opérateur "return" doit rendre la valeur
122	Une déclaration incorrecte du destructeur
123	Les deux-points manquent ":"
124	La variable est déjà déclarée
125	La variable avec un tel identificateur est déjà déclarée
126	Le nom de la variable est trop long (>250 caractères)
127	La structure avec un tel identificateur est déjà définie
128	La structure n'est pas définie
129	Le membre de la structure avec un tel nom est déjà défini
130	Il n'y a pas de tel membre de la structure
131	La paire des crochets est troublée
132	La parenthèse ouvrante est attendue "("
133	Les accolades déséquilibrées (manque "}")
134	Difficile pour la compilation (le branchage est trop grand, la pile interne des niveaux est remplis)
135	L'erreur de l'ouverture du fichier à la lecture
136	Pas assez de mémoire pour le chargement du fichier initial dans la mémoire
137	La variable est attendue
138	La référence ne peut pas être initialisé
140	On s'attendait l'affectation (apparaît à la déclaration)
141	L'accolade ouvrante est attendue "{"
142	Le paramètre peut être seulement un tableau dynamique
143	L'utilisation du type "void" est inadmissible
144	Il n'y a pas de paire pour ")" ou "]", c'est- à- dire "(" ou "[" manquent
145	Il n'y a pas de paire pour "(" ou "[",c'est- à- dire ")" ou "]" manquent
146	La dimension incorrecte du tableau

© 2000-2015, MetaQuotes Software Corp.

147	Il y a trop de paramètres (>64)
149	Ce jeton n'est pas prévu ici
150	L'utilisation inadmissible de l'opération (les opérandes incorrects)
151	L'expression du type "void" est inadmissible
152	L'opérateur est attendu
153	Une mauvaise utilisation de "break"
154	Le point-virgule est attendu ";"
155	La virgule est attendue ","
156	Le type doit être défini comme la classe, et pas comme la structure
157	L'expression est attendue
158	A HEX se rencontre "non HEX caractère" ou un trop long nombre (la quantité de chiffres > 511)
159	La ligne-constante a plus de 65534 caractères
160	La définition de la fonction est inadmissible ici
161	Une fin inattendue du programme
162	Une déclaration préalable pour les structures est interdite
163	La fonction avec un tel nom est déjà définie et a un autre type de la valeur rendue
164	La fonction avec un tel nom est déjà définie et a un autre ensemble des paramètres
165	La fonction avec un tel nom est déjà définie et réalisée
166	La surcharge de la fonction pour l'appel donné n'est pas trouvée
167	La fonction avec une valeur rendue comme "void" ne peut pas rendre la valeur
168	La fonction n'est pas définie
170	La valeur est attendue
171	Dans l'expression "case" sont admissibles seulement les constantes entières
172	La valeur pour "case" dans ce "switch" est déjà utilisée
173	Une valeur entière est attendue

© 2000-2015, MetaQuotes Software Corp.

174	Dans l'expression #import le nom du fichier est attendu
175	Les expressions au niveau global ne sont pas admissibles
176	La parenthèse ")" est manquée devant ";"
177	La variable se trouve à gauche du signe d'égalité
178	Le résultat de l'expression n'est pas utilisé
179	La déclaration des variables dans "case" est inadmissible
180	La transformation implicite de la ligne au nombre
181	La transformation implicite du nombre à la ligne
182	L'appel ambigu de la fonction surchargée (plusieurs surcharges conviennent)
183	"else" est inadmissible sans "if" correspondant
184	"case" ou "default" est inadmissible sans "switch" correspondant
185	L'utilisation inadmissible de l'ellipse
186	L'héritage initialisant a plus de quantité d'éléments que la variable initialisée
187	La constante est attendue pour "case"
188	On demande l'expression constante
189	La variable constante ne peut pas être changée
190	La parenthèse fermante ou la virgule est attendue (la déclaration du membre du tableau)
191	L'identificateur de l'énumération est déjà utilisé
192	L'énumération ne peut pas avoir les modificateurs de l'accès (const, extern, static)
193	Le membre de l'énumération est déjà déclaré avec une autre valeur
194	Il y a une variable définie avec le même nom
195	Il y a une structure définie avec le même nom
196	Le nom du membre de l'énumération est attendu
197	L'expression entière est attendue

© 2000-2015, MetaQuotes Software Corp.

198	La division en zéro dans l'expression constante
199	La quantité incorrecte de paramètres à la fonction
200	Le paramètre selon le lien doit être la variable
201	La variable du même type pour la transmission selon le lien est attendue
202	La variable constante ne peut pas être transmise selon le lien non constante
203	On demande la constante positive entière
204	L'erreur de l'accès au membre protégé de la classe
205	L'importation est déjà définie selon un autre chemin
208	Le fichier exécuté n'est pas créé
209	On ne trouve pas le point de l'entrée 'OnCalculate' pour l'indicateur
210	L'opérateur continue peut être utilisé seulement à l'intérieur de la boucle
211	L'erreur de l'accès au "private" (fermé) de la classe
213	La méthode de la structure ou la classe n'est pas déclarée
214	L'erreur de l'accès au "private" (fermé) de la méthode de classe
216	Le copiage des structures avec les objets est inadmissible
218	La sortie de l'index pour les frontières du tableau
219	L'initialisation des tableaux dans la déclaration de la structure ou la classe est inadmissible
220	Le constructeur de la classe ne peut pas avoir les paramètres
221	Le décomposeur de la classe ne peut pas avoir les paramètres
222	La méthode de la classe ou la structure avec un tel nom et les paramètres est déjà déclarée
223	L'opérande est attendue
224	Il y a la méthode de la classe ou la structure avec un tel nom, mais avec les autres

© 2000-2015, MetaQuotes Software Corp.

	paramètres (la déclaration! = la réalisation)
225	La fonction importée n'est pas décrite
226	Функция ZeroMemory() не применима для классов с защищенными членами или наследованием
227	L'appel ambigu de la fonction surchargée (la coïncidence exacte des paramètres pour quelques surcharges)
228	Le nom de la variable est attendu
229	On ne peut pas déclarer le lien dans cette place
230	Est utilisé déjà à titre du nom del'énumération
232	La classe ou la structure est attendue
235	Il est interdit d'appeler "delete" pour supprimer le tableau
236	L'opérateur ' while ' est attendu
237	Un indicateur doit être dans "delete"
238	il y a "switch" pour ce "default"
239	L'erreur syntaxique
240	Escape-l'héritage peut se rencontrer seulement dans les lignes (commence par '\')
241	Le tableau est necéssaire - le crochet '[' ne s'applique pas au tableau, ou non le tableau est passé comme comme paramètre-tableau
242	Ne peut pas être initialisé au moyen de l'héritage initialisant
243	L'importation n'est pas définie
244	L'erreur de l'optimisateur sur l'arbre syntaxique
245	On a déclaré trop de structures (simplifiez le programme)
246	La transformation du paramètre est inadmissible
247	L'utilisation incorrecte de l'opérateur "delete"
248	On ne peut pas déclarer l'indicateur sur la référence
249	On ne peut pas déclarer une référence à une référence
250	On ne peut pas déclarer l'indicateur à l'indicateur

© 2000-2015, MetaQuotes Software Corp.

251	Il est inadmissible de déclarer la structure dans la liste des paramètres
252	L'opération inadmissible de la réduction des types
253	On peut déclarer l'indicateur seulement pour la classe ou la structure
256	L'identificateur non déclaré
257	L'erreur de l'optimiseur du code exécuté
258	L'erreur de la génération du code exécuté
260	L'expression inadmissible pour l'opérateur "switch"
261	Le débordement du pool des constantes de la chaîne, simplifiez le programme
262	Il est impossible de transcrire l'énumération
263	On ne peut pas utiliser "virtual" pour les données (les membres de la classe ou la structure)
264	On ne peut pas provoquer la méthode protégée de la classe
265	La fonction redéfinie virtuelle rend un autre type
266	On ne peut pas hériter la classe de la structure
267	On ne peut pas hériter la structure de la classe
268	Le constructeur ne peut pas être virtuel (le spécificateur virtual est inadmissible)
269	La structure ne peut pas avoir les méthodes virtuelles
270	La fonction doit avoir le corps
271	La surcharge des fonctions systémiques (les fonctions du terminal) est interdite
272	Le spécificateur "const" est inadmissible aux fonctions lesquelles ne sont pas les membres de la classe ou la structure
274	On ne peut pas changer les membres de la classe dans la méthode constante
276	L'héritage inconvenant initialisant
277	On manque la valeur par défaut pour le paramètre (la spécificité de la déclaration des paramètres par défaut)

© 2000-2015, MetaQuotes Software Corp.

278	La redéfinition du paramètre par défaut (dans la déclaration et la réalisation des différentes valeurs)
279	On ne peut pas provoquer la méthode non constante pour l'objet constant
280	On demande l'objet pour l'accès aux membres (on met le point pour non de la classe/ structure)
281	On ne peut pas utiliser le nom la structure déjà déclarée dans la déclaration
284	La conversion faite sans autorisation (à l'héritage fermé)
285	Les structures et les tableaux ne peuvent pas être utilisées à titre des variables "input"
286	Le spécificateur "const" est inadmissible au constructeur/destructeur
287	L'expression incorrecte de chaîne pour le type "datetime"
288	La propriété inconnue (#property)
289	La valeur incorrecte pour la propriété
290	La valeur incorrecte pour la propriété dans #property
291	Le paramètre de l'appel manque - < func(x,) >
293	L'objet doit être transmis selon le lien
294	Le tableau doit être transmis selon le lien
295	La fonction était déclarée comme exporté
296	La fonction n'était pas déclarée comme exporté
297	Il est interdit d'exporter la fonction importée
298	La fonction importée ne peut pas avoir un tel paramètre (on ne peut pas transmettre l'indicateur, la classe ou la structure contenant le tableau dynamique, l'indicateur, la classe etc.)
299	La classe doit être
300	La section #import n'est pas fermée
302	La non-conformité des types
303	La variable extern est déjà initialisée
304	N'est pas trouvé aucune fonction exportée ou le

© 2000-2015, MetaQuotes Software Corp.

	point standard de l'entrée
305	L'appel évident du constructeur est interdit
306	La méthode a été déclarée constante
307	La méthode n'a pas été déclarée constante
308	La taille incorrecte du fichier de ressource
309	Le nom incorrect de la ressource
310	L'erreur de l'ouverture du fichier de la ressource
311	L'erreur de la lecture du fichier de la ressource
312	Le type incorrect de la ressource
313	Un chemin incorrect vers le fichier de la ressource
314	Le nom indiqué de la ressource est déjà utilisé
315	on attendait les paramètres de la macro
316	Après le nom de la macro doit être l'espace
317	L'erreur dans la description des paramètres de la macro
318	Le nombre incorrecte des paramètres à l'utilisation de la macro
319	Le dépassement du nombre maximal (16) des paramètres de la macro
320	La macro est trop compliquée, il faut la simplifier
321	Seulement l'énumération peut êtrele paramètre EnumToString()
322	Le nom de la ressource est trop trop long
323	Le format non soutenu de l'image (le format BMP avec la profondeur de la couleur de 24 ou 32 bits est seulement admissible)
324	La déclaration du tableau à l'intérieur de l'opérateur est interdite
325	On peut définir la fonction seulement sur le niveau global
326	Cette déclaration est inadmissible pour le domaine de la visibilité actuel (le domaine de la définition)
327	L'initialisation des variables statiques par les valeurs des variables locaux est inadmissible

© 2000-2015, MetaQuotes Software Corp.

328	La déclaration inadmissible du tableau des objets qui n'ont pas le constructeur par défaut
329	La liste de l'initialisation est permise seulement pour les constructeurs
330	La définition de la fonction après la liste de l'initialisation manque
331	La liste de l'initialisation est vide
332	L'initialisation du tableau dans le constructeur est interdite
333	Il est interdit d'initialiserles membres de la classe parentale dans la liste de l'initialisation
334	L'expressiondu type entiera été attendu
335	Le volume demandé de la mémoire pourle tableau dépasse la valeur au maximum admissible
336	Le volume demandé de la mémoire pour la structure dépasse la valeur au maximum admissible
337	Le volume demandé de la mémoire pour les variables, déclarées sur le niveau global, dépasse la valeur au maximum admissible
338	Le volume demandé de la mémoire pour les variables locales dépasse la valeur au maximum admissible
339	Le constructeur n'est pas défini
340	Le nom inadmissible pour le fichier de l'icône
341	On n'a pas réussi à ouvrir le fichier de l'icôneselon un chemin spécifié
342	Le fichier de l'icône est incorrect et ne correspond pas au format ICO
343	L'initialisation réitérée du membre dans le constructeur de la classe/structure à l'aide de la liste de l'initialisation
344	L'initialisationdes membres statiquesdans la liste de l'initialisationdu constructeur n'est pas admis
345	L'initialisationdu membre/structure non statique auniveau global est interdite
346	Le nom de la méthode de la classe/structure coïncide avec le nom auparavant annoncé du

© 2000-2015, MetaQuotes Software Corp.

	membre
347	Le nom du membre de la classe/structure coïncide avec le nom auparavant annoncé de la méthode
348	La fonctionvirtuelle ne peut pas être annoncée commestatic
349	Le modificateur const est inadmissible pourla fonction statique
350	Le constructeur ou destructeur_ ne peuvent pas être statiques
351	On ne peut pas s'adresser au membre/méthode non statique de la classe ou la structure de la fonction statique
352	Après le mot clef operator l'opération surchargée est attendue (+,-,[],++,-- etc.)
353	Pas toutes les opérations peuvent être surchargées dans MQL5
354	La définition ne correspond pas à la déclaration
355	On a indiqué la quantité incorrecte de paramètres pour l'opérateur
356	Aucune fonction qui traite l'événement n'est pas découverte
357	Les méthodes ne peuvent pas être exportées
358	On ne peut pas amener l'indice à l'objetconstant à l'indice à l'objet non constant
359	Les modèles des classes ne sont pas encore soutenus
360	La surchargedes modèles des fonction n'est pas encore soutenue
361	Il est impossible d'appliquer le modèle de la fonction
362	Le paramètre ambigu dans le modèle de la fonction (quelques types du paramètre conviennent)
363	Il est impossible de définir vers quel type du paramètre amener l'argument du modèle de la fonction
364	Le nombre incorrecte des paramètres dans un modèle de la fonction
365	Le modèle de la fonction ne peut pas êtrevirtuel

© 2000-2015, MetaQuotes Software Corp.

366	Les modèles des fonctions ne peuvent pas être exportés
367	On ne peut pas importer les modèles des fonctions
368	Les structures contenant des objets sont inadmissibles
369	Массивы строк и структуры, содержащие объекты, недопустимы
370	Статический член класса/структуры должен быть явно инициализирован
371	Ограничение компилятора: строка не может содержать более 65 535 символов
372	Несогласованные #ifdef/#endif
373	Результатом выполнения функции не может быть объект класса, так как отсутствует конструктор копирования
374	Нельзя использовать нестатические члены и/или методы при инициализации статической переменной
375	OnTesterInit() нельзя использовать без объявления обработчика OnTesterDeinit()
376	Имя локальной переменной совпадает с именем одного из параметров функции
377	Нельзя использовать макросы __FUNCSIG__ и __FUNCTION__ вне тела функции
378	Invalid returned type. For example, this error will be produced for functions imported from DLL that return structure or pointer.

© 2000-2015, MetaQuotes Software Corp.

Les erreurs du temps de l'exécution

GetLastError() – La fonction rendant le code de la dernière erreur, qui se trouve dans la variable prédéterminée_LastError. On peut enlever la valeur de cette variable au zéro par la fonction ResetLastError().

Constante	Valeur	Description
ERR_INTERNAL_ERROR	4001	L'erreur inattendue intérieure
ERR_WRONG_INTERNAL_PARAMETER	4002	Le paramètre faux à l'appel intérieur de la fonction du terminal de client
ERR_INVALID_PARAMETER	4003	Le paramètre faux à l'appel intérieur de la fonction systémique
ERR_NOT_ENOUGH_MEMORY	4004	Il ne suffit pas la mémoire pour l'exécution de la fonction systémique
ERR_STRUCT_WITHOBJECTS_ORCLASS	4005	La structure contient les objets des lignes et/ou les tableaux dynamiques et/ou la structure avec tels objets et/ou les classes
ERR_INVALID_ARRAY	4006	Le tableau du type inconvenant, de la grandeur inconvenante ou u l'objet abîmé du tableau dynamique
ERR_ARRAY_RESIZE_ERROR	4007	Il ne suffit pas la mémoire pour la redistribution du tableau ou la tentative du changement de la grandeur du tableau statique
ERR_STRING_RESIZE_ERROR	4008	Il ne suffit pas la mémoire pour la redistribution de la ligne
ERR_NOTINITIALIZED_STRING	4009	La ligne n'est pas initialisée
ERR_INVALID_DATETIME	4010	La valeur incorrecte de la date et/ou le temps
ERR_ARRAY_BAD_SIZE	4011	La grandeur de tableau demandé excède 2 gigoctets
ERR_INVALID_POINTER	4012	L'indicateur faux
ERR_INVALID_POINTER_TYPE	4013	Le type de l'indicateur faux
ERR_FUNCTION_NOT_ALLOWE	4014	La fonction systémique n'est

© 2000-2015, MetaQuotes Software Corp.

D		pas autorisé pour l'appel
ERR_RESOURCE_NAME_DUPLICATED	4015	a coïncidence des nomsdes ressources et dynamique etstatique
ERR_RESOURCE_NOT_FOUND	4016	La ressource avec un tel nom à EX5 n'est pas trouvée
ERR_RESOURCE_UNSUPPOTED_TYPE	4017	Le type non soutenu de la ressource ou la taille plus 16 Mb
ERR_RESOURCE_NAME_IS_TOO_LONG	4018	Le nom de la ressource dépasse 63 symboles
Graphiques		
ERR_CHART_WRONG_ID	4101	L'indificateur faux de chart
ERR_CHART_NO_REPLY	4102	Le chart ne répond pas
ERR_CHART_NOT_FOUND	4103	Le chart n'est pas trouvé
ERR_CHART_NO_EXPERT	4104	Le chart n'a pas d'expert, qui pourrait traiter l'événement
ERR_CHART_CANNOT_OPEN	4105	L'erreur de l'ouverture de chart
ERR_CHART_CANNOT_CHANGE	4106	L'erreur au changement de chart du symbole et la période
ERR_CHART_WRONG_PARAMETER	4107	La valeur erronée du paramètre pour la fonction du travail avec le graphique
ERR_CHART_CANNOT_CREATE_TIMER	4108	L'erreur à la création du minuteur
ERR_CHART_WRONG_PROPERTY	4109	L'identificateur faux de la propriété du chart
ERR_CHART_SCREENSHOT_FAILED	4110	L'erreur à la création des screenshots
ERR_CHART_NAVIGATE_FAILED	4111	L'erreur de la navigation dans le chart
ERR_CHART_TEMPLATE_FAILED	4112	L'erreur à l'application de la modèle
ERR_CHART_WINDOW_NOT_FOUND	4113	La sous- fenêtre contenant l'indicateur indiqué, n'est pas trouvé
ERR_CHART_INDICATOR_CANNOT_ADD	4114	L'erreur à l'ajout de l'indicateur sur le graphique
ERR_CHART_INDICATOR_CANN	4115	L'erreur à la suppression de

© 2000-2015, MetaQuotes Software Corp.

OT_DEL		l'indicateur du graphique
ERR_CHART_INDICATOR_NOT_FOUND	4116	L'indicateur n'est pas trouvé sur le graphique indiqué
Les objets graphiques		
ERR_OBJECT_ERROR	4201	L'erreur au fonctionnement de l'objet graphique
ERR_OBJECT_NOT_FOUND	4202	L'objet graphique n'est pas trouvé
ERR_OBJECT_WRONG_PROPERTY	4203	L'identificateur faux de la propriété de l'objet graphique
ERR_OBJECT_GETDATE_FAILED	4204	Il est impossible de recevoir la date correspondant à la valeur
ERR_OBJECT_GETVALUE_FAILED	4205	Il est impossible de recevoir la valeur correspondante à la date
MarketInfo		
ERR_MARKET_UNKNOWN_SYMBOL	4301	Le symbole inconnu
ERR_MARKET_NOT_SELECTED	4302	Le symbole n'est pas choisi dans MarketWatch
ERR_MARKET_WRONG_PROPERTY	4303	L'identificateur faux de la propriété du symbole
ERR_MARKET_LASTTIME_UNKNOWN	4304	Le temps du dernier tique est inconnu (il n'y avait pas de ticks)
ERR_MARKET_SELECT_ERROR	4305	L'erreur de l'ajout ou de la suppression du symbole dans MarketWatch
L'accès à l'histoire		
ERR_HISTORY_NOT_FOUND	4401	L'histoire demandée n'est pas trouvée
ERR_HISTORY_WRONG_PROPERTY	4402	L'identificateur faux de la propriété de l'histoire
Global_Variables		
ERR_GLOBALVARIABLE_NOT_FOUND	4501	La variable globale du terminal de client n'est pas trouvée
ERR_GLOBALVARIABLE_EXISTS	4502	La variable globale du terminal de client avec un tel nom existe déjà

© 2000-2015, MetaQuotes Software Corp.

ERR_MAIL_SEND_FAILED	4510	On ne réussit pas à expédier le message
ERR_PLAY_SOUND_FAILED	4511	On ne réussit pas à reproduire le son
ERR_MQL5_WRONG_PROPERTY	4512	L'identificateur faux de la propriété du programme
ERR_TERMINAL_WRONG_PROPERTY	4513	L'identificateur faux de la propriété du terminal
ERR_FTP_SEND_FAILED	4514	On ne réussit pas à expédier le fichier selon ftp
ERR_NOTIFICATION_SEND_FAILED	4515	On n'a pas réussi d'envoyer la notification
ERR_NOTIFICATION_WRONG_PARAMETER	4516	Le paramètre non valide pour envoyer une notification - on a transmis une chaîne vide dans une fonction SendNotification() ou NULL
ERR_NOTIFICATION_WRONG_SETTINGS	4517	Les réglages incorrectes des notifications dans le terminal (ID n'est pas indiqué ou la permission n'est pas exposée)
ERR_NOTIFICATION_TOO_FREQUENT	4518	L'expédition trop fréquente des notifications
Les tampons des indicateurs d'utilisateur		
ERR_BUFFERS_NO_MEMORY	4601	Il ne suffit pas la mémoire pour la distribution des tampons d'indicateur
ERR_BUFFERS_WRONG_INDEX	4602	L'index faux du tampon d'indicateur
Les propriétés des indicateurs d'utilisateur		
ERR_CUSTOM_WRONG_PROPERTY	4603	L'identificateur faux de la propriété de l'indicateur d'utilisateur
Account		
ERR_ACCOUNT_WRONG_PROPERTY	4701	L'identificateur faux de la propriété du compte
ERR_TRADE_WRONG_PROPERTY	4751	L'identificateur faux de la propriété du commerce
ERR_TRADE_DISABLED	4752	Le commerce pour l'expert est

© 2000-2015, MetaQuotes Software Corp.

		interdit
ERR_TRADE_POSITION_NOT_FOUND	4753	La position n'est pas trouvée
ERR_TRADE_ORDER_NOT_FOUND	4754	L'ordre n'est pas trouvée
ERR_TRADE_DEAL_NOT_FOUND	4755	Le marché n'est pas trouvé
ERR_TRADE_SEND_FAILED	4756	On n'a pas réussi à expédier la requête commerciale
Indicateurs		
ERR_INDICATOR_UNKNOWN_SYMBOL	4801	Le symbole inconnu
ERR_INDICATOR_CANNOT_CREATE	4802	L'indicateur ne peut pas être créé
ERR_INDICATOR_NO_MEMORY	4803	Pas assez de mémoire pour ajouter l'indicateur
ERR_INDICATOR_CANNOT_APPLY	4804	L'indicateur ne peut pas être appliqué à un autre indicateur
ERR_INDICATOR_CANNOT_ADD	4805	L'erreur à l' incrémentation de l'indicateur
ERR_INDICATOR_DATA_NOT_FOUND	4806	Les données demandées ne sont pas trouvées
ERR_INDICATOR_WRONG_INDEX	4807	Le handle de l'indicateur erroné
ERR_INDICATOR_WRONG_PARAMETERS	4808	La quantité incorrecte de paramètres à la création de l'indicateur
ERR_INDICATOR_PARAMETERS_MISSING	4809	Les paramètres manquent à la création de l'indicateur
ERR_INDICATOR_CUSTOM_NAME	4810	Le premier paramètre dans le tableau doit être le nom de l'indicateur d'utilisateur
ERR_INDICATOR_PARAMETER_TYPE	4811	Le type incorrect du paramètre dans le tableau à la création de l'indicateur
ERR_INDICATOR_WRONG_INDEX	4812	L'index erroné du tampon d'indicateur demandé
Profondeur de marché		
ERR_BOOKS_CANNOT_ADD	4901	Le profondeur de marché ne peut pas être ajouté

© 2000-2015, MetaQuotes Software Corp.

ERR_BOOKS_CANNOT_DELETE	4902	Le profondeur de marché ne peut pas être supprimé
ERR_BOOKS_CANNOT_GET	4903	Les données du profondeur de marché ne peuvent pas être reçues
ERR_BOOKS_CANNOT_SUBSCRIBE	4904	L'erreur à la souscription sur la réception des nouvelles données du profondeur de marché
Opérations de fichier		
ERR_TOO_MANY_FILES	5001	Ne peut pas être ouvert simultanément plus de 64 fichiers
ERR_WRONG_FILENAME	5002	Le nom inadmissible du fichier
ERR_TOO_LONG_FILENAME	5003	Un trop long nom du fichier
ERR_CANNOT_OPEN_FILE	5004	L'erreur de l'ouverture du fichier
ERR_FILE_CACHEBUFFER_ERROR	5005	Pas assez de mémoire pour le cache pour la lecture
ERR_CANNOT_DELETE_FILE	5006	L'erreur de l'effacement du fichier
ERR_INVALID_FILEHANDLE	5007	Le fichier avec un tel handle a été fermé, ou ne s'ouvrait pas du tout
ERR_WRONG_FILEHANDLE	5008	Le handle faux du fichier
ERR_FILE_NOTTOWRITE	5009	Le fichier doit être ouvert pour l'enregistrement
ERR_FILE_NOTTOREAD	5010	Le fichier doit être ouvert pour la lecture
ERR_FILE_NOTBIN	5011	Le fichier doit être ouvert comme binaire
ERR_FILE_NOTTXT	5012	Le fichier doit être ouvert comme le texte
ERR_FILE_NOTTXTORCSV	5013	Le fichier doit être ouvert comme le texte ou CSV
ERR_FILE_NOTCSV	5014	Le fichier doit être ouvert comme CSV
ERR_FILE_READERROR	5015	L'erreur de la lecture du fichier
ERR_FILE_BINSTRINGSIZE	5016	La dimension de chaîne doit être spécifiée, puisque le

© 2000-2015, MetaQuotes Software Corp.

		fichier est ouvert comme binaire
ERR_INCOMPATIBLE_FILE	5017	Pour les tableaux de ligne doit être un fichier de texte, pour les autres - le fichier binaire
ERR_FILE_IS_DIRECTORY	5018	Ce n'est pas un fichier, c'est un répertoire
ERR_FILE_NOT_EXIST	5019	Le fichier n'existe pas
ERR_FILE_CANNOT_REWRITE	5020	Le fichier ne peut pas être réécrit
ERR_WRONG_DIRECTORYNAME	5021	Le nom faux du répertoire
ERR_DIRECTORY_NOT_EXIST	5022	Le répertoire n'existe pas
ERR_FILE_ISNOT_DIRECTORY	5023	C'est un fichier, pas un répertoire
ERR_CANNOT_DELETE_DIRECTORY	5024	Le répertoire ne peut pas être supprimé
ERR_CANNOT_CLEAN_DIRECTORY	5025	On n'a pas réussi à vider le répertoire (probablement, un ou quelques fichiers sont bloqués et l'opération de l'effacement n'a pas réussi)
ERR_FILE_WRITEERROR	5026	On n'a pas réussi à enregistrer la ressource au fichier
Conversion des chaînes		
ERR_NO_STRING_DATE	5030	Il n'y a pas de date dans la chaîne
ERR_WRONG_STRING_DATE	5031	Une date fausse dans la chaîne
ERR_WRONG_STRING_TIME	5032	Un temps faux dans la chaîne
ERR_STRING_TIME_ERROR	5033	L'erreur de la conversion de la chaîne à la date
ERR_STRING_OUT_OF_MEMORY	5034	Pas assez de mémoire pour la chaîne
ERR_STRING_SMALL_LEN	5035	La longueur de chaîne est moins qu'attendue
ERR_STRING_TOO_BIGNUMBER	5036	Le nombre est trop grand, plus que ULONG_MAX
ERR_WRONG_FORMATSTRING	5037	La chaîne fausse de format
ERR_TOO_MANY_FORMATTER	5038	Plus de spécificateurs de

© 2000-2015, MetaQuotes Software Corp.

S		format que des paramètres
ERR_TOO_MANY_PARAMETERS	5039	Plus de paramètres que des spécificateurs de format
ERR_WRONG_STRING_PARAMETER	5040	Le paramètre abîmé du type string
ERR_STRINGPOS_OUTOFRANGE	5041	La position en dehors de la chaîne
ERR_STRING_ZEROADDED	5042	0 est ajouté à la fin de la chaîne, l'opération est inutile
ERR_STRING_UNKNOWNTYPE	5043	Le type de données inconnu à la conversion à la chaîne
ERR_WRONG_STRING_OBJECT	5044	L'objet abîmé de la chaîne
Opérations avec les tableaux		
ERR_INCOMPATIBLE_ARRAYS	5050	Le copiage des tableaux incompatibles. Le tableau de ligne peut être copié seulement à celui de ligne, et le tableau numérique – à celui numérique
ERR_SMALL_ASSERIES_ARRAY	5051	Le tableau de la réception est déclaré comme AS_SERIES, et il est de la taille insuffisante
ERR_SMALL_ARRAY	5052	Un trop petit tableau, la position de départ est à l'extérieur du tableau
ERR_ZEROSIZE_ARRAY	5053	Le tableau de la longueur nulle
ERR_NUMBER_ARRAYS_ONLY	5054	Le tableau doit être numérique
ERR_ONEDIM_ARRAYS_ONLY	5055	Le tableau doit être unidimentionnel
ERR_SERIES_ARRAY	5056	La série temporelle ne peut pas être utilisée
ERR_DOUBLE_ARRAY_ONLY	5057	Le tableau doit être du type double
ERR_FLOAT_ARRAY_ONLY	5058	Le tableau doit être du type float
ERR_LONG_ARRAY_ONLY	5059	Le tableau doit être du type long
ERR_INT_ARRAY_ONLY	5060	Le tableau doit être du type int
ERR_SHORT_ARRAY_ONLY	5061	Le tableau doit être du type

© 2000-2015, MetaQuotes Software Corp.

		short
ERR_CHAR_ARRAY_ONLY	5062	Le tableau doit être du type char
Le travail avec OpenCL		
ERR_OPENCL_NOT_SUPPORTED	5100	Les fonctions OpenCL ne sont pas soutenues sur cet ordinateur
ERR_OPENCL_INTERNAL	5101	l'erreur intérieure à l'exécution d' OpenCL
ERR_OPENCL_INVALID_HANDLE	5102	Le handleincorrect OpenCL
ERR_OPENCL_CONTEXT_CREATE	5103	L'erreur à la création du contexte OpenCL
ERR_OPENCL_QUEUE_CREATE	5104	L'erreur de la création du tour de l'exécution dans OpenCL
ERR_OPENCL_PROGRAM_CREATE	5105	L'erreur pendant la compilation du programme OpenCL
ERR_OPENCL_TOO_LONG_KERNEL_NAME	5106	Un trop long nom du point de l'entrée (le kernel OpenCL)
ERR_OPENCL_KERNEL_CREATE	5107	L'erreur de la créationdu point-kernel de l'entrée OpenCL
ERR_OPENCL_SET_KERNEL_PARAMETER	5108	L'erreur à l'installation des paramètres pour le kernel OpenCL (le point de l'entrée au programme OpenCL)
ERR_OPENCL_EXECUTE	5109	L'erreur de l'exécution du programme OpenCL
ERR_OPENCL_WRONG_BUFFER_SIZE	5110	Une taille incorrecte du tampon OpenCL
ERR_OPENCL_WRONG_BUFFER_OFFSET	5111	Un décalage incorrect dans le tampon OpenCL
ERR_OPENCL_BUFFER_CREATE	5112	L'erreur de la création du tampon OpenCL
Работа с WebRequest		
ERR_WEBREQUEST_INVALID_ADDRESS	5200	URL не прошел проверку
ERR_WEBREQUEST_CONNECT_FAILED	5201	Не удалось подключиться к указанному URL
ERR_WEBREQUEST_TIMEOUT	5202	Превышен таймаут получения данных

© 2000-2015, MetaQuotes Software Corp.

ERR_WEBREQUEST_REQUEST_FAILED	5203	Ошибка в результате выполнения HTTP запроса
Erreurs d'utilisateur		
ERR_USER_ERROR_FIRST	65536	Par ce code commencent les erreurs, spécifiées par l'utilisateur

Voir aussi

Codes du retour du serveur commercial

© 2000-2015, MetaQuotes Software Corp.

Les constantes de l'entrée/sortie

Les constantes:

- Les drapeaux de l'ouverture des fichiers
- Propriétés des fichiers
- Le positionnement à l'intérieur des fichiers
- L'utilisation de la page de code
- MessageBox

© 2000-2015, MetaQuotes Software Corp.

Les drapeaux de l'ouverture des fichiers

Les valeurs des drapeaux définissant le mode de travail avec le fichier. Les drapeaux sont définis comme ça:

Identificateur	Valeur	Description
FILE_READ	1	Le fichier s'ouvre pour la lecture. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()). A l'ouverture du fichier on doit indiquer absolument le drapeau FILE_WRITE et/ou le drapeux FILE_READ.
FILE_WRITE	2	Le fichier s'ouvre pour l'enregistrement. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()). A l'ouverture du fichier on doit indiquer absolument le drapeau FILE_WRITE et/ou le drapeux FILE_READ.
FILE_BIN	4	Le mode binaire de la lecture-enregistrement (sans conversion de la chaîne et à la chaîne). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())
FILE_CSV	8	Le fichier du type csv (Tous les éléments inscrits seront transcrits vers les chaînes du type correspondant, unicode ou ansi, et se divisent par le séparateurU). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())
FILE_TXT	16	Un simple fichier de texte (le même csv, cependant le séparateur n'est pas pris en considération). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())
FILE_ANSI	32	Les chaînes du type ANSI (les symboles d'un byte). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())
FILE_UNICODE	64	Les chaînes du type UNICODE

© 2000-2015, MetaQuotes Software Corp.

		(les symboles de deux bytes). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())
FILE_SHARE_READ	128	L'accès commun selon la lecture de plusieurs programmes. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()), mais ne remplace pas à l'ouverture du fichier la nécessité d'indiquer FILE_WRITE et/ou le drapeau FILE_READ
FILE_SHARE_WRITE	256	L'accès commun selon l'enregistrement de plusieurs programmes. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()), mais ne remplace pas à l'ouverture du fichier la nécessité d'indiquer FILE_WRITE et/ou le drapeau FILE_READ
FILE_REWRITE	512	La possibilité de réécrire le fichier par les fonctions FileCopy() et FileMove(). Le fichier doit exister ou s'ouvrir pour l'enregistrement. Dans le cas contraire le fichier ne sera pas ouvert
FILE_COMMON	4096	La disposition du fichier dans le dossier commun de tous les terminaux de client \Terminal \Common\Files. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()), au copiage des fichiers (FileCopy(), FileMove()) et à la vérification de l'existence des fichiers (FileIsExist())

A l'ouverture du fichier on peut indiquer un ou plus des drapeaux, une telle combinaison s'appelle la combinaison des drapeaux. La combinaison des drapeaux s'inscrit à l'aide du signe de l'opération logique OU (|), qui est mis entre les drapeaux énumérés. Par exemple pour ouvrir le fichier dans le format CSV simultanément sur la lecture et sur l'enregistrement, on peut indiquer la combinaison FILE_READ|FILE_WRITE|FILE_CSV.

Exemple:

```
int filehandle=FileOpen(filename,FILE_READ|FILE_WRITE|FILE_CSV);
```

Il y a certaines particularités du travail à l'indication des drapeaux de la lecture et de l'enregistrement:

© 2000-2015, MetaQuotes Software Corp.

- Si FILE_READ est indiqué- on fait la tentative de l'ouverture du fichier existant. Si le fichier n'existe pas, on ne réussira pas à ouvrir le fichier, un nouveau fichier n'est pas créé.
- Si FILE_READ|FILE_WRITE - on crée un nouveau fichier s'il n' y a pas de fichier avec un tel nom.
- Si FILE_WRITE - on crée du nouveau le fichier avec la grandeur zéro.

À l'ouverture du fichier on doit indiquer absolument le drapeau FILE_WRITE et/ou le drapeux FILE_READ.

Les drapeaux définissant le type de la lecture du fichier ouvert, ont la priorité. Le drapeau le plus principal FILE_CSV, après suit le drapeau FILE_BIN selon la majorité, et la plus petite priorité a le drapeau FILE_TXT. Ainsi, si soudain on indique à la fois quelques drapeaux (FILE_TXT|FILE_CSV ou FILE_TXT|FILE_BIN ou FILE_BIN|FILE_CSV), le drapeau le plus grand selon la priorité sera utilisé.

Les drapeaux, définissant le type de codage ont aussi la priorité. Le drapeau FILE_UNICODE ont la plus grande priorité que le drapeau FILE_ANSI. C'est pourquoi à l'indication de la combinaison FILE_UNICODE|FILE_ANSI on utilisera le drapeu FILE_UNICODE.

Si n'est pas indiqué ni FILE_UNICODE, ni FILE_ANSI, il est sousentendu FILE_UNICODE. Si n'est pas indiqué ni FILE_CSV, ni FILE_BIN, ni FILE_TXT, il est sousentendu FILE_CSV.

Si le fichier est ouvert pour la lecture comme celui de texte (FILE_TXT ou FILE_CSV), et de plus au début du fichier est découvert l'indication spéciale `0xff,0xfe` de 2 bytes, le drapeux du codage sera FILE_UNICODE, même si on a indiqué le drapeau FILE_ANSI.

Voir aussi

Les opérations des fichiers

© 2000-2015, MetaQuotes Software Corp.

Propriétés des fichiers

Pour la réception des propriétés des fichiers on utilise la fonction FileGetInteger(), auquelle on transmet l'identificateur de la propriété demandée de l'énumération ENUM_FILE_PROPERTY_INTEGER à l'appel

ENUM_FILE_PROPERTY_INTEGER

Identificateur	La description de l'identificateur
FILE_EXISTS	La vérification de l'existence
FILE_CREATE_DATE	La date de la création
FILE_MODIFY_DATE	La date du dernier changement
FILE_ACCESS_DATE	La date du dernier accès au fichier
FILE_SIZE	La taille du fichier en bytes
FILE_POSITION	La position du pointeur au fichier
FILE_END	La réception du critère de la fin du fichier
FILE_LINE_END	La réception du critère de la fin de la ligne
FILE_IS_COMMON	Le fichier est ouvert dans un dossier commun de tous les terminaux de client (à voir FILE_COMMON)
FILE_IS_TEXT	Le fichier est ouvert comme celui de texte (à voir FILE_TXT)
FILE_IS_BINARY	Le fichier est ouvert comme binaire (à voir FILE_BIN)
FILE_IS_CSV	Le fichier est ouvert comme CSV (à voir FILE_CSV)
FILE_IS_ANSI	Le fichier est ouvert comme ANSI (à voir FILE_ANSI)
FILE_IS_READABLE	Le fichier est ouvert avec la possibilité de la lecture (à voir FILE_READ)
FILE_IS_WRITABLE	Le fichier est ouvert avec la possibilité de l'enregistrement (à voir FILE_WRITE)

La fonctionFileGetInteger() a deux variantes de l'appel. Dans une première variante pour la réception des propriétés du fichier est indiqué son handle, reçu à l'ouverture du fichier par la fonction FileOpen(). Cette variante permet de recevoir toutes les propriétés du fichier.

La deuxième variante de la fonction FileGetInteger() rend les valeurs des propriétés du fichier selon son nom. Dans cette variante on peut recevoir seulement les propriétés communes suivantes:

- FILE_EXISTS – l'existence du fichier indiqué selon le nom;
- FILE_CREATE_DATE – la date de la création du fichier avec le nom indiqué;

© 2000-2015, MetaQuotes Software Corp.

- FILE_MODIFY_DATE – la date du changement du fichier avec le nom indiqué;
- FILE_ACCESS_DATE – la date du dernier accès au fichier avec le nom indiqué;
- FILE_SIZE – la taille du fichier avec le nom indiqué.

A la tentative de la réception des autres propriétés, sauf indiquées ci-dessus, la deuxième variante de l'appel de la fonction FileGetInteger() rendra l'erreur.

© 2000-2015, MetaQuotes Software Corp.

Le positionnement à l'intérieur du fichier

La grande partie des fonctions de fichier est liée aux opérations la lecture/inscription de l'information. En cela à l'aide de la fonction FileSeek() on peut indiquer la position de l'indicateur de fichier sur la position à l'intérieur du fichier, de laquelle l'opération suivante de la lecture ou l'enregistrement sera produite. L'énumération ENUM_FILE_POSITION contient les positions admissibles de l'indicateur, relativement auxquelles on peut indiquer les décalages dans les bytes pour l'opération suivante.

ENUM_FILE_POSITION

Identificateur	Description
SEEK_SET	Le début du fichier
SEEK_CUR	La position courante de l'indicateur de fichier
SEEK_END	La fin du fichier

Voir aussi

FileIsEnding, FileIsLineEnding

© 2000-2015, MetaQuotes Software Corp.

L'utilisation de la page de code dans les opérations de la conversion des chaînes

Aux opérations de la conversion des variables de chaîne aux tableux du type char et à l'inverse dans le langage MQL5 on utilise le codage, correspondant par défaut au codage ANSI courant du système opérationnel Windows (CP_ACP). S'il faut indiquer un autre type du codage, on peut le donner par le paramètre supplémentaire pour les fonctions CharArrayToString(), StringToCharArray() et FileOpen().

Le tableau énumère les constantes intégrées pour certaines des pages codées les plus populaires. On peut indiquer les pages non énumérées de code par le code correspondant à cette page.

Les constantes insérées des pages de code

Constante	Valeur	Description
CP_ACP	0	Une page de code courant ANSI le codage au système opérationnel Windows
CP_OEMCP	1	Une page de code courant OEM.
CP_MACCP	2	Une page de code courant Macintosh. **Note:** Cette signification est utilisée principalement dans les codes de programme auparavant créés et maintenant on n'a pas besoin de lui, puisque les ordinateurs modernes Macintosh utilisent le codage Unicode.
CP_THREAD_ACP	3	Le codage Windows ANSI pour le thread courant d'exécution.
CP_SYMBOL	42	La page de code Symbol
CP_UTF7	65000	La page de code UTF-7.
CP_UTF8	65001	La page de code UTF-8.

Voir aussi

L'état du terminal de client

© 2000-2015, MetaQuotes Software Corp.

Les constantes de fenêtre de dialogue MessageBox

Les codes de retour de la fonction MessageBox(). Si la fenêtre du message la Suppression (Cancel), la fonction rend la valeur IDCANCEL au bouton appuyé ESC ou le bouton la Suppression (Cancel). Si la fenêtre du message n'a pas de bouton la Suppression (Cancel), la pression de ESC ne donne pas aucun effet.

Constante	Valeur	Description
IDOK	1	On a choisi le bouton OK
IDCANCEL	2	On a choisi le bouton Suppression (Cancel)
IDABORT	3	On a choisi le bouton Interrompre (Abort)
IDRETRY	4	On a choisi le bouton Répétition (Retry)
IDIGNORE	5	On a choisi le bouton Ignorer (Ignore)
IDYES	6	On a choisi le bouton Oui (Yes)
IDNO	7	On a choisi le bouton No (No)
IDTRYAGAIN	10	On a choisi le bouton Répéter (Try Again)
IDCONTINUE	11	On a choisi le bouton Continuer (Continue)

Les drapeaux principaux de la fonction MessageBox() définissent le contenu et le comportement de la boîte de dialogue. Cette valeur peut être la combinaison des drapeaux des groupes suivants des drapeaux:

Constante	Valeur	Description
MB_OK	0x00000000	La fenêtre du message contient un bouton: OK. Par défaut
MB_OKCANCEL	0x00000001	La fenêtre du message contient deux boutons: OK et Cancel
MB_ABORTRETRYIGNORE	0x00000002	La fenêtre du message contient trois boutons: Abort, Retry et Ignore
MB_YESNOCANCEL	0x00000003	La fenêtre du message contient trois boutons: Yes, No

© 2000-2015, MetaQuotes Software Corp.

		et Cancel
MB_YESNO	0x00000004	La fenêtre du message contient deux boutons: Yes et No
MB_RETRYCANCEL	0x00000005	La fenêtre du message contient deux boutons: Retry et Cancel
MB_CANCELTRYCONTINUE	0x00000006	La fenêtre du message contient trois boutons: Cancel, Try Again, Continue

Pour afficher une icône dans la fenêtre de message il est nécessaire de spécifier des drapeaux supplémentaires:

Constante	Valeur	Description
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	0x00000010	L'icône de signe STOP
MB_ICONQUESTION	0x00000020	L'icône de signe de question
MB_ICONEXCLAMATION, MB_ICONWARNING	0x00000030	L'icône de signe de question
MB_ICONINFORMATION, MB_ICONASTERISK	0x00000040	L'icône, comprenant le signe de ligne **i** dans le cercle

Les boutons sont définis par défaut par les drapeaux suivants:

Constante	Valeur	Description
MB_DEFBUTTON1	0x00000000	Le premier bouton MB_DEFBUTTON1 - le bouton est choisi par défaut, si MB_DEFBUTTON2, MB_DEFBUTTON3, ou MB_DEFBUTTON4 ne sont pas définis
MB_DEFBUTTON2	0x00000100	Le deuxième bouton - le bouton par défaut
MB_DEFBUTTON3	0x00000200	Le troisième bouton - le bouton par défaut
MB_DEFBUTTON4	0x00000300	Le quatrième bouton - le bouton par défaut

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

Les programmes MQL5

Pour que le programme mql5 puisse fonctionner elle doit être compilée (le bouton "Compiler" ou la touche F7). La compilation doit passer sans erreurs (les préventions sont admissibles, il faut les analyser). En cela dans le répertoire correspondant, *terminal_dir\MQL5\Experts*, *terminal_dir\MQL5\indicators* ou *terminal_dir\MQL5\scripts*, doit être créé un fichier exécuté avec le même nom et l'extension EX5. Justement ce fichier peut être lancé sur l'exécution.

Les particularités du travail des programmes mql5 sont décrites dans les paragraphes:

- L'exécution des programmes – l'ordre de l'appel des fonctions prédéterminées - des gestionnaires des événements;
- Le test des stratégies commerciales – les particularités du travail des programmes dans le testeur des stratégies;
- Les événements du terminal de client – la description des événements, dont on peut traiter dans les programmes;
- L'appel des fonctions importées – les ordres de la description, les paramètres admissibles, l'ordre de la recherche et l'accord sur les liens des fonctions importées;
- Les erreurs de l'exécution – la réception de l'information sur les erreurs de l'exécution et les erreurs critiques.

Les experts, les indicateurs d'utilisateur et les scripts se fixent à un des graphiques ouverts par la méthode Drag'n'Drop de la fenêtre "le Navigateur" du terminal de client sur le graphique correspondant (la technologie Drag'n'Drop). Les programmes mql5 peuvent fonctionner seulement au terminal de client branché.

Pour que l'expert cesse de travailler, il est nécessaire de le supprimer du graphique en sélectionnant "Les Conseillers - Supprimer" du menu contextuel du graphique. L'état du bouton "Permettre/interdire les conseillers" influence au travail du conseiller.

Pour que l'indicateur d'utilisateur cesse le travail, il est nécessaire de le supprimer du graphique.

Les indicateurs d'utilisateur et les conseillers travaillent jusqu'à ce qu'ils soient explicitement supprimés d'un graphique; l'information sur les conseillers attachés et les indicateurs d'utilisateur est sauvée entre les lancements du terminal de client.

Les scripts sont exécutés une fois et se suppriment automatiquement à la fin du travail ou de la clôture ou le changement de l'état du graphique courant ou à la fin du travail du terminal de client. A la relance du terminal de client les scripts ne sont pas lancés, puisque l'information sur eux n'est pas sauvée.

Maximum un expert, un script et un nombre illimité d'indicateurs peuvent travailler dans un graphique.

© 2000-2015, MetaQuotes Software Corp.

L'exécution des programmes

Chaque script et chaque expert travaille dans son propre thread séparé. Tous les indicateurs calculés sur un symbole travaillent dans un thread, même s'ils sont lancés sur les graphiques différents. Ainsi tous les indicateurs sur un symbole divisent entre eux-mêmes les ressources d'un thread.

Aussi dans un thread avec les indicateurs on accomplit successivement les autres actions du symbole donné - le traitement des ticks et la synchronisation de l'histoire. Cela signifie que si l'action infinie est accomplie dans l'indicateur, tous les autres événements selon son symbole ne seront jamais accomplis.

Au lancement de l'expert il faut assurer à lui environnement commercialactuel, l'accessibilité de l'histoire selon le symbole donné et la période, ainsi que produire la synchronisation entre le terminal et le serveur. Sur ces procédures le terminal accorde le délai du lancement à l'expert pas plus qu'à 5 secondes, après quoi l'expert sera lancé avec ces données qu'on a réussi à préparer. Voilà pourquoi en absence de communication avec le serveur, ça peut retarder le lancement de l'expert.

Un bref résumé selon les programmes MQL5 est amené dans un tableau:

Programme	Exécution	Note
Script	Dans son propre thread, tant de scripts existent - autant de threads d'exécution il y a pour eux.	Le script avec une boucle infinie ne peut pas perturber le travail des autres programmes
Expert	Dans son propre thread, tant d'experts existent - autant d'experts d'exécution il y a pour eux.	L'expert avec une boucle infinie ne peut pas perturber le travail des autres programmes
Indicateur	Le thread de l'exécution pour tous les indicateurs sur un symbole. Tant de symboles avec indicateurs existent - autant de threads d'exécution il y a pour eux	Une boucle infinie dans un indicateur arrêtera le travail de tous les autres indicateurs sur ce symbole

Directement après qu'un programme est attaché à un graphique on produit son chargement à la mémoire du terminal de client et l'initialisation des variables globales. Si quelque variable globale comme la classe a le constructeur, cet constructeur sera appellé en train de l'initialisation des variables globales.

Après cela le programme se trouve dans l'état de l'attente de l'événement du terminal de client. Chaque programme mql5 avoir être quand même une fonction-gestionnaire de l'événement, dans le cas contraire programme chargé ne sera pas accompli. Les fonctions-gestionnaires des événements ont les noms prédéterminés, les ensembles prédéterminés des paramètres et les types prédéterminés du retour.

Type	Nom de la fonction	Paramètres	Application	Commentaires

© 2000-2015, MetaQuotes Software Corp.

int	OnInit	non	Experts et Indicateurs	Le gestionnaire de l'événement Init. On admet le type de la valeur rendue void.
void	OnDeinit	const int reason	Experts et Indicateurs	Le gestionnaire de l'événement Deinit.
void	OnStart	non	Scripts	Le gestionnaire de l'événement Start.
int	OnCalculate	const int rates_total, const int prev_calculated, const datetime &Time[], const double &Open[], const double &High[], const double &Low[], const double &Close[], const long &TickVolume[], const long &Volume[], const int &Spread[]	Indicateurs	Le gestionnaire de l'événement Calculate sur toutes les données de prix.
int	OnCalculate	const int rates_total, const int prev_calculated, const int begin, const double &price[]	Indicateurs	Le gestionnaire de l'événement Calculate sur un tableau des données. Dans l'indicateur ne peut pas simultanément assister 2 gestionnaires Calculate. Dans ce cas travaillera seulement un gestionnaire de l'événement Calculate sur un tableau des données.

© 2000-2015, MetaQuotes Software Corp.

void	OnTick	non	Experts	Le gestionnaire de l'événement NewTick. Pendant le traitement de l'événement de l'arrivée du nouveau tick les autres événements de ce type ne viennent pas.
void	OnTimer	non	Experts et Indicateurs	Le gestionnaire de l'événement Timer.
void	OnTrade	non	Experts	Le gestionnaire de l'événement Trade.
void	OnTester	non	Experts	Le gestionnaire de l'événement Tester.
void	OnChartEvent	const int id, const long &lparam, const double &dparam, const string &sparam	Experts et Indicateurs	Le gestionnaire de l'événement ChartEvent.
void	OnBookEvent	const string &symbol_name	Experts et Indicateurs	Le gestionnaire de l'événement BookEvent.

Le terminal de client envoie les événements apparaissant aux graphiques correspondants ouverts. Aussi les événements peuvent être générés par les graphiques (les événements du graphique) ou par les programmes mql5 (les événements d'utilisateur). On peut activer et désactiver la génération des événements de la création et de la suppression des objets graphiques sur le graphique en spécifiant des propriétés du graphique CHART_EVENT_OBJECT_CREATE et CHART_EVENT_OBJECT_DELETE. Chaque programme MQL5 et chaque graphique a sa propre file d'attente d'événements, où sont ajoutés tous les nouveaux événements entrants.

Le programme reçoit les événements seulement du graphique, où il est lancé. Tous les événements sont traités l'un après l'autre dans l'ordre reçu. Si l'événement NewTick se trouve déjà dans la file d'attente ou cet événement est en état du traitement, on ne met pas un nouvel événement NewTick dans la file d'attente du programme mql5. Analogiquement, si l'événement ChartEventest déjà dans la file d'attente du programme mql5 ou cet événement est en état du traitement, on ne met pas un nouvel événement de ce type dans la file d'attente. Le traitement des événements de la minuterie est produit selon le même schéma – si l'événement Timerest au tour ou est déjà traité, un nouvel événement de la minuterie n'est pas mis au tour.

© 2000-2015, MetaQuotes Software Corp.

Les tours des événements ont la taille limitée, mais suffisante, c'est pourquoi le débordement pour les programmes correctement écrits est peu probable. Au débordement du tour les nouveaux événements sont rejetés sans mise dans la file d'attente.

Il n'est pas recommandé d'utiliser des boucles infinies pour traiter les événements. Seulement les scripts qui traitent un seul événement Start peuvent être l'exception de cette règle.

Les bibliothèques ne traitent pas aucuns événements.

L'interdiction à l'utilisation des fonctions dans les indicateurs et les experts

Les indicateurs, les scripts et les experts sont les programmes exécutés de MQL5 et sont destinés aux différents types des tâches. C'est pourquoi il y a une limitations sur l'utilisation des fonctions définies en fonction de type du programme. Les fonctions suivantes sont interdites dans les indicateurs:

- OrderCalcMargin();
- OrderCalcProfit();
- OrderCheck();
- OrderSend();
- SendFTP();
- Sleep();
- ExpertRemove();
- MessageBox().

A son tour, toutes les fonctions, destinées aux indicateurs sont interdites dans les experts et les scripts:

- SetIndexBuffer();
- IndicatorSetDouble();
- IndicatorSetInteger();
- IndicatorSetString();
- PlotIndexSetDouble();
- PlotIndexSetInteger();
- PlotIndexSetString();
- PlotIndexGetInteger.

La bibliothèque n'est pas le programme indépendant et elle s'exécute dans le contexte du programme MQL5 qui l'a appelée: le script, l'indicateur ou l'expert. En conséquence, toutes les limitations indiquées ci-dessus se répandent à la bibliothèque appelée.

Le chargement et le déchargement des indicateurs

Les indicateurs sont chargés dans les cas suivants :

- l'indicateur est attaché à un graphique;

© 2000-2015, MetaQuotes Software Corp.

- le lancement du terminal (si l'indicateur était attaché au graphique avant la clôture du terminal);
- le chargement d'un modèle (si l'indicateur attaché à un graphique est spécifié dans le modèle);
- le changement d'un profil (si l'indicateur est attaché à un des graphiques de profil);
- le changement d'un symbole et/ou la période du graphique, auquel l'indicateur est attaché;
- après la recompilation réussie d'un indicateur, si cet indicateur était attaché à un graphique.
- le changement des paramètres d'entrée de l'indicateur.

Les indicateurs sont déchargés dans les cas suivants :

- en détachant un indicateur du graphique;
- la clôture du terminal (si l'indicateur était attaché au graphique);
- le chargement d'un modèle, si un indicateur est attaché au graphique ;
- la fermeture d'un graphique, auquel l'indicateur a été attaché;
- le changement d'un profil, si l'indicateur est attaché à un de graphiques du profil changé;
- le changement d'un symbole et/ou la période du graphique, auquel l'indicateur est attaché;
- le changement les paramètres d'entrée de l'indicateur.

Le chargement et le déchargement des experts

L'expert est chargé dans les cas suivants:

- un expert est attaché à un graphique;
- le lancement du terminal (si l'expert était attaché au graphique avant la clôture du terminal);
- le chargement d'un modèle (si l'expert attaché au graphique est spécifié dans le modèle);
- après la recompilation réussie d'un expert, si cet expert était attaché à un graphique;
- le changement d'un profil, si l'expert est attaché à un de graphiques du profil changé);
- la connexion au compte, même si le numéro du compte n'a pas changé (si l'expert a été fixé au graphique avant l'autorisation du terminal sur le serveur).

Le déchargement de l'expert attaché au graphique, est produit dans les cas suivants:

- en détachant un expert du graphique;
- si un nouvel Expert est attaché à un graphique, si un autre expert a été déjà attaché, cet expert est déchargé;
- la clôture du terminal (si l'expert était attaché au graphique);
- le chargement d'un modèle, si un expert est attaché au graphique;
- la clôture d'un graphique, auquel l'expert est attaché;
- le changement d'un profil, si l'expert est attaché à un de graphiques du profil changé;
- le changement du compte, auquel le terminal est connecté (si l'expert a été fixé au graphique avant l'autorisation du terminal sur le serveur);
- calling the ExpertRemove() function.

Au changement du caractère ou du temps trame du graphique, auquel l'expert est attaché le déchargement et le chargement de l'expert n'est pas produit. Dans ce cas les gestionnaires

© 2000-2015, MetaQuotes Software Corp.

OnDeinit() sont appelés successivement sur un ancien symbole/temps trame et OnInit() au nouveau symbole/temps trame (s'ils existent), les valeurs des variables globales et des variables statiques ne sont pas réinitialisées. Tous les événements qui sont entrés pour l'expert jusqu'à l'achèvement de l'initialisation (les fonctions OnInit()) se sont manqués.

Le chargement et le déchargement des scripts

Les scripts sont chargés immédiatement après qu'ils sont attachés à un graphique et déchargé immédiatement après qu'ils accomplissent leur opération. En cela les fonctions OnInit() et OnDeinit() ne sont pas appelées pour les scripts.

Au déchargement du programme (supprimer le programme du graphique) se passe la déinitialisation des variables globales et l'annulation du tour des messages. Dans ce cas la déinitialisation signifie la désallocation des variables du type string, la libération des objets des tableaux dynamiques et l'appel des destructeurs à leur présence.

Pour la meilleure compréhension du travail de l'expert il est recommandé de compiler le code de l'expert amené dans l'exemple et produire les actions du chargement/déchargement des experts, le remplacement de la modèle, le caractère, le temps trame etc.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                                   TestExpert.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

class CTestClass
  {
public:
   CTestClass() { Print("CTestClass constructor"); }
   ~CTestClass() { Print("CTestClass destructor"); }
  };
CTestClass global;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("Initialisation");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print("Deinitialisation with reason ",reason);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
```

Les scripts sont chargés directement après avoir fixé le graphique et déchargé immédiatement après la fin de ses travaux.

Voir aussi

Les événements du terminal de client, Les fonctions de traitement des événements

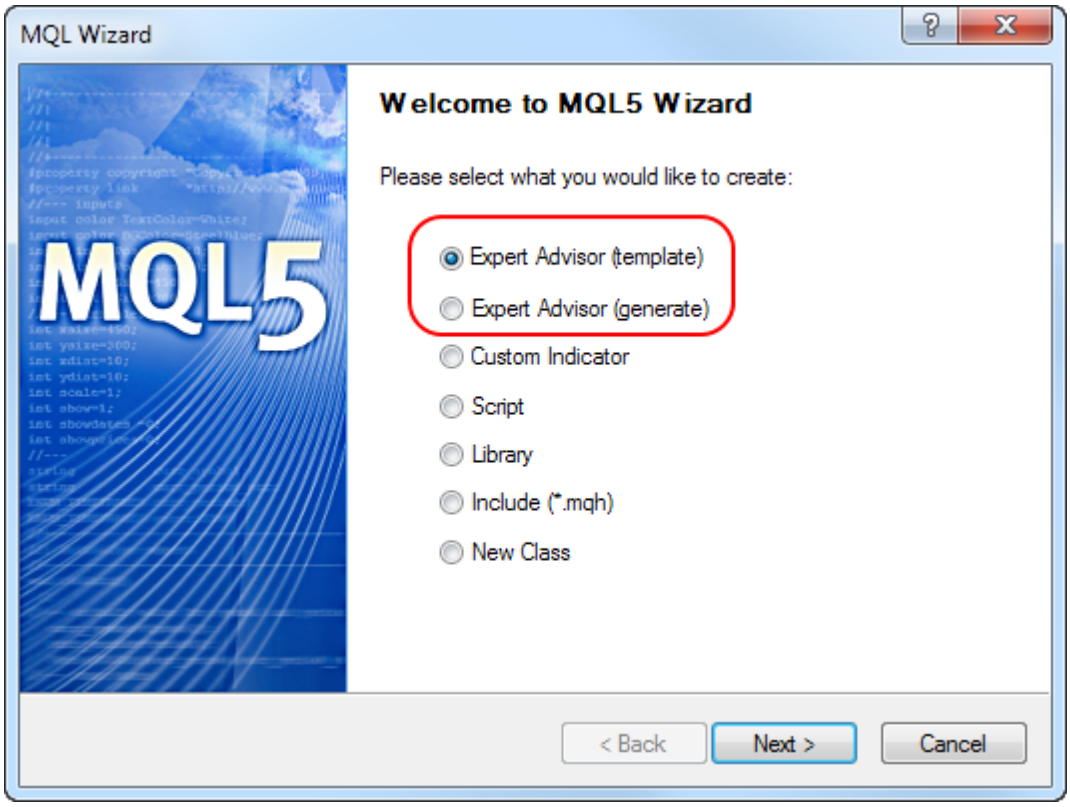
© 2000-2015, MetaQuotes Software Corp.

Разрешение на торговлю

Автоматизация торговли

В языке MQL5 существует специальная группа торговых функций, с помощью которых можно создавать автоматизированные торговые системы. Программы для автоматической торговли без участия человека называются экспертами (Expert Advisor) или торговыми роботами. Для создания эксперта в редакторе MetaEditor запустите мастер создания советников MQL5 Wizard и выберите один из двух вариантов:

- Expert Advisor (template) - позволяет создать шаблон с готовыми функциями обработки событий, который необходимо дополнить всем необходимым функционалом с помощью самостоятельного программирования.
- Expert Advisor (generate) - дает возможность создать полностью готового для работы торгового робота, просто выбирая необходимые вам модули: модуль формирования торговых сигналов, модуль управления капиталом и модуль подтягивания защитного стопа для открытых позиций (Trailing Stop).



Торговые функции могут работать только в экспертах и скриптах, для индикаторов торговля запрещена.

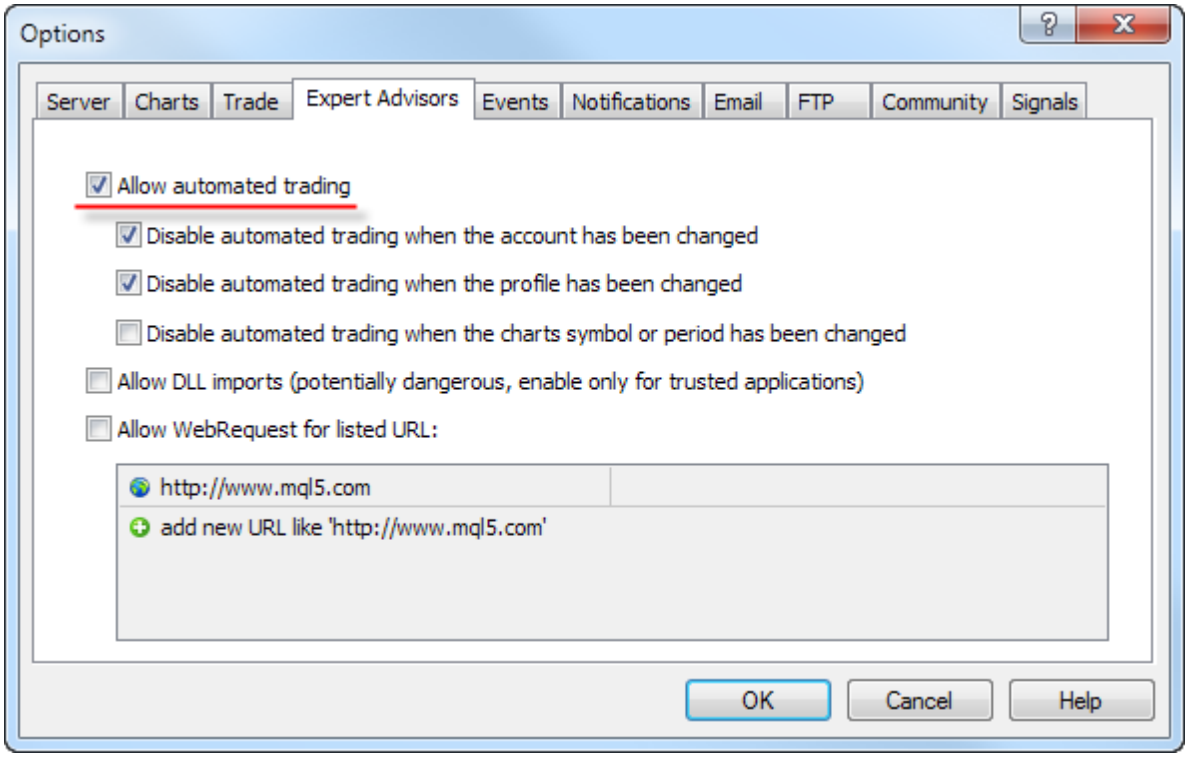
Проверка разрешения на автоматическую торговлю

Для создания надежного эксперта, который мог бы работать в автоматическом режиме без контроля человека, необходимо организовать множество необходимых проверок. В первую очередь необходимо программным путем проверять, есть ли вообще разрешение на совершение торговых операций. Проверка разрешения на торговлю является базовой и обязательной при разработке любой автоматической системы.

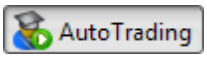
© 2000-2015, MetaQuotes Software Corp.

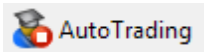
Проверка на разрешение автоматической торговли в самом терминале

В настройках терминала можно запретить или разрешить автоматическую торговлю для всех программ.



Переключать опцию разрешения автоматической торговли также можно прямо из Стандартной панели терминала:

- 

- автоматическая торговля разрешена, торговые функции в запущенных программах разрешены для использования.
- 

- автоматическая торговля запрещена, при этом сами запущенные программы будут работать, хотя торговые функции не смогут выполняться.

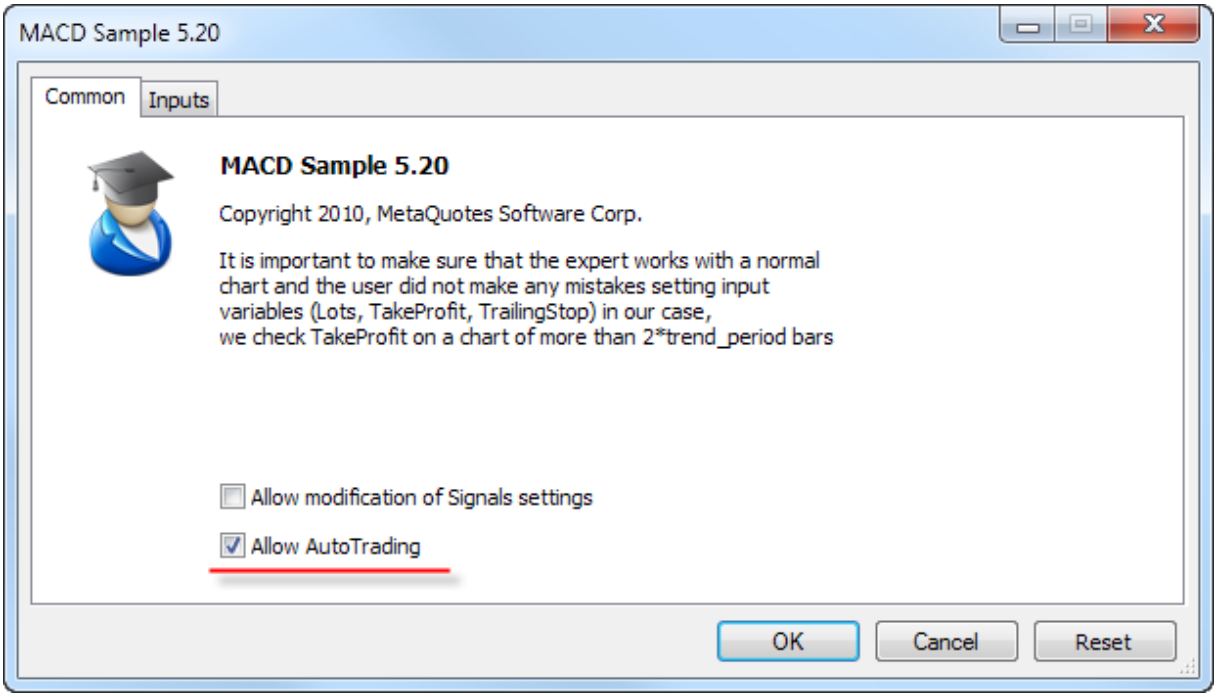
Пример проверки:

```
if (!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("Проверьте в настройках терминала разрешение на автоматическую торговлю!");
```

Проверка разрешения на торговлю для данного запущенного эксперта/скрипта

При запуске программы можно разрешить или запретить автоматическую торговлю конкретно для нее. Для этого есть отдельная специальная настройка в свойствах программы.

© 2000-2015, MetaQuotes Software Corp.



Пример проверки:

```
if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("Проверьте в настройках терминала разрешение на автоматическую торговлю!")
else
  {
   if(!MQLInfoInteger(MQL_TRADE_ALLOWED))
      Alert("Автоматическая торговля запрещена в свойствах программы для ",__FILE__
  }
```

Проверка разрешения на торговлю любым экспертам/скриптам для данного счета

Запрет на автоматическую торговлю может быть установлен на стороне торгового сервера. Пример проверки такой ситуации:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_EXPERT))
   Alert("Автоматическая торговля запрещена для счета ",AccountInfoInteger(ACCOUNT_
   " на стороне торгового сервера");
```

Если для торгового счета запрещена автоматическая торговля, то торговые операции из экспертов/скриптов выполняться не будут.

Проверка разрешения торговли для данного счета

Возможны случаи, когда для конкретного торгового счета запрещены любые торговые операции - нельзя торговать ни вручную, ни с помощью экспертов. Пример проверки ситуации, когда к торговому счету подключились с помощью инвесторского пароля:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
   Comment("Торговля запрещена для счета ",AccountInfoInteger(ACCOUNT_LOGIN),
```

© 2000-2015, MetaQuotes Software Corp.

```
            ".\n Возможно, подключение к торговому счету произведено по инвест паролю.
            "\n Проверьте журнал терминала, есть ли там такая запись:",
            "\n\'",AccountInfoInteger(ACCOUNT_LOGIN),"\': trading has been disabled -
```

AccountInfoInteger(ACCOUNT_TRADE_ALLOWED) может возвращать false в следующих случаях:

- нет соединения с торговым сервером. Можно проверить с помощью TerminalInfoInteger(TERMINAL_CONNECTED);
- торговый счет переведен в режим read-only (отправлен в архив);
- торговля на счете запрещена на стороне торгового сервера;
- подключение к торговому счете произведено в режиме инвестора.

Смотри также

Состояние клиентского терминала, Информация о счете, Информация о запущенной MQL5-программе

© 2000-2015, MetaQuotes Software Corp.

Les événements du terminal de client

Init

Tout de suite après que le terminal de client chargera le programme (l'expert ou l'indicateur d'utilisateur) et lancera le procès de l'initialisation des variables globales, l'événement Init, sera envoyé, qui sera traité par la fonction OnInit(), si elle existe. Cet événement est généré aussi après le remplacement de l'instrument financier et/ou la période du graphique, après la recompilation du programme dans le rédacteur MetaEditor, après le changement des paramètres d'entrée de la fenêtre du réglage de l'expert ou l'indicateur d'utilisateur. Le conseiller est initialisé aussi après le changement du compte. Pour les scripts l'événement Init ne se généré pas.

Deinit

Devant la déinitialisation des variables globales et le déchargement du programme (l'expert ou l'indicateur d'utilisateur) le terminal de client envoie au programme l'événement Deinit. L'événement Deinit est généré aussi à l'achèvement du travail du terminal de client, à la clôture du graphique, directement devant le changement de l'instrument financier et/ou la période du graphique, à la recompilation réussie du programme, au changement des paramètres d'entrée, ainsi qu'au changement du compte.

La_raison_de_deinitialisation peut être obtenue du paramètre, transmis à la fonction OnDeinit(). L'exécution de la fonction OnDeinit () se limite à 2.5 secondes. Si pour ce temps la fonction n'a pas fini le travail, son exécution s'achève forcément. Pour les scripts l'événement Deinit ne se généré pas.

Start

L'événement Start – est un événement spécial pour l'activation de script après son chargement. Cet événement est traité par la fonction OnStart. L'événement Start n'est pas envoyé aux experts et aux indicateurs d'utilisateur.

NewTick

L'événement NewTick est généré à l'entrée des nouvelles cotations et est traité par la fonction OnTick() chez les conseillers adjoints. Si à l'entrée de la nouvelle cotation on accomplissait la fonctio OnTick, lancée sur la cotation précédente, la cotation reçue sera ignorée par le conseiller, puisque l'événement correspondant n'est pas mis au tour des événements de l'expert.

Toutes les nouvelles cotations qui sont venues pendant l'exécution du programme sont ignorées par le programme jusqu'à ce que s'achève l'exécution suivante de la fonction OnTick(). Après cela la fonction sera lancée seulement après l'arrivée de la nouvelle cotation suivante. L'événement NewTick n'est pas généré à l'interdiction insérée de l'utilisation des conseillers (le bouton "Permettre/interdire les conseillers").

L'événement NewTick est généré indépendamment du fait, si le commerce automatique est permis ou interdit (le bouton "Permettre/interdire le commerce automatique"). L'interdiction du commerce automatique signifie seulement l'interdiction à l'expédition des requêtes commerciales de l'expert, le travail de l'expert ne cesse pas.

L'interdiction du commerce automatique par voie de la pression sur le bouton indiqué n'interrompt pas l'exécution de la fonction courrante OnTick().

© 2000-2015, MetaQuotes Software Corp.

Calculate

L'événement Calculate est généré seulement pour les indicateurs après l'envoi de l'événement Init et à n'importe quel changement des données de prix. Il est traité par la fonction OnCalculate.

Timer

L'événement Timer est généré périodiquement par le terminal de client pour l'expert, qui a activé le minuteur à l'aide de la fonction EventSetTime. D'habitude cette fonction est appelé dans une fonction OnInit. Le traitement de l'événement Timer est produit par la fonction OnTimer. A l'achèvement du travail de l'expert il est nécessaire de supprimer le minuteur créé l'aide de EventKillTimer, qui est d'habitude appelé à la fonction OnDeinit.

Trade

L'événement Trade est généré à l'achèvement de l'opération commerciale sur le serveur commercial. Le traitement de l'événement Trade est produit par la fonction OnTrade() pour les opérations suivantes commerciales:

- L'installation, la modification ou l'effacement de l'ordre remis;
- l'annulation de l'ordre remis au manque des moyens ou à l'expiration;
- le fonctionnement de l'ordre remis;
- l'ouverture, le supplément ou la clôture de la position (ou de la partie de la position);
- la modification de la position ouverte (le changement des Stops).

TradeTransaction

A la suite de l'exécution des actions définies avec le compte commercial, son état change. Ces actions comprennent:

- L'envoi de la demande commerciale par n'importe quelle application MQL5- dans le terminal de client à l'aide des fonctions OrderSend et OrderSendAsync et son exécution ultérieure;
- L'envoi de la demande commerciale par l'interface graphique du terminal et son exécution ultérieure;
- le fonctionnement des ordres remis et des ordres stop sur le serveur;
- L'exécution des opérations sur le côté du serveur commercial.

A la suite de ces actions, les transactions commerciales sont exécutées pour le compte:

- le traitement de la demande commerciale;
- le changement des ordres ouverts;
- le changement de l'histoire des ordres;
- le changement de l'histoire des marchés;
- le changement des positions.

Par exemple, à l'envoi de l'ordre de marché sur l'achat, il est traité, l'ordre correspondant sur l'achat est créé pour le compte, se passe l'exécution de l'ordre, son suppression de la liste des ouverts, l'ajout à l'histoire des ordres, ensuite le marché correspondant est ajouté à l'histoire et une nouvelle position est créé. Toutes ces actions sont les transactions commerciales. L'arrivée de chaque telle transaction au terminal est l'événement TradeTransaction. Cet événement est traité par la fonctionOnTradeTransaction.

© 2000-2015, MetaQuotes Software Corp.

Tester

L'événement Tester est généré à l'achèvement du test de l'expert des données historiques. Le traitement de l'événement Tester est produit par la fonction OnTester().

TesterInit

L'événement TesterInit est généré au lancement de l'optimisation dans le testeur des stratégies avant le premier passage. Le traitement de l'événement TesterInit est produit par la fonctionOnTesterInit().

TesterPass

L'événementTesterPass est généré à l'entréed'une nouvelle trame des données. Le traitement de l'événement TesterPass est produit par la fonction OnTesterPass().

TesterDeinit

L'événement TesterDeinit est généré à l'issue de l'optimisation de l'expert dans le testeur des stratégies. Le traitement de l'événement TesterDeinit est produit par la fonctionOnTesterDeinit().

ChartEvent

L'événement ChartEvent est généré par le terminal de client pendant le travail d'utilisateur avec le graphique:

- la pression du clavier, quand la fenêtre du graphique se trouve dans le tour;
- la création de l'objet graphique;
- l'effacement l'objet graphique;
- le clic de la souris sur l'objet graphique, appartenant au graphique;
- le déplacement de l'objet graphique à l'aide de la souris;
- la fin de l'édition du texte dans le champ d'entrée de l'objet graphique LabelEdit.

Il y a aussi un événement d'utilisateur ChartEvent, qui peut être envoyé à l'expert par chaque programme mql5 à l'aide de la fonction EventChartCustom. L'événement est traité par la fonction OnChartEvent.

BookEvent

L'événement BookEvent est généré par le terminal de client au changement de l'état du profondeur de marché et est traité par la fonction OnBookEvent. Pour que le terminal de client commence à générer les événements BookEvent selon le symbole concret, c'est assez de s'inscrire préalablement sur la réception de ces événements pour ce symbole à l'aide de la fonction MarketBookAdd.

Pour désabonner de la réception de l'événement BookEvent selon le symbole, il est nécessaire d'appeler la fonction MarketBookRelease. L'événement BookEvent est celui de diffusion – cela signifie qu'il est suffisant à un expert de souscrire à la réception de l'événement BookEvent à l'aide de la fonction MarketBookAdd, et tous les autres experts qui ont le gestionnaire OnBookEvent, recevrons cet événement. C'est pourquoi il est nécessaire d'analyser le nom du symbole, qui est transmis au gestionnaire à titre du paramètre.

Voir aussi

© 2000-2015, MetaQuotes Software Corp.

Les fonctions du traitement des événements, L'exécution des programmes

© 2000-2015, MetaQuotes Software Corp.

Ressources

Utilisation du graphique et du son dans les programmes sur MQL5

Les programmes sur MQL5 permettent de travailler avec les fichiers sonores et graphiques:

- PlaySound() joue un fichier sonore;
- ObjectCreate() permet de créer les interfaces d'utilisateur à l'aidedes objets graphiques OBJ_BITMAP et OBJ_BITMAP_LABEL.

PlaySound()

Exemple de l'appel de la fonction PlaySound ():

```
//+------------------------------------------------------------------+
//| La fonction appelle OrderSend() titulaire et joue le son         |
//+------------------------------------------------------------------+
void OrderSendWithAudio(MqlTradeRequest  &request, MqlTradeResult &result)
  {
  //--- envoyons la demande au serveur
   OrderSend(request,result);
  //--- si la demande est acceptée, jouons un son Ok.wav
   if(result.retcode==TRADE_RETCODE_PLACED) PlaySound("Ok.wav");
   //--- à l'échec nous jouons le son alarmant du fichier timeout.wav
   else PlaySound("timeout.wav");
  }
```

Cet exemple montre comment jouer les sons à partir des fichiers Ok.wav et timeoit.wav, qui entrent dans la livraison standard du terminal. Ces fichiers se trouvent dans le dossier **le répertoire_du terminal\Sounds**. Ici **le répertoire_du terminal** signifie le dossier à partir duquel vous avez exécuté le terminal de client MetaTrader 5. Par la voie programmatique du programme mql5 on peut savoir le répertoire du terminal comme il suit:

```
//--- Le dossier, où se trouvent les données du terminal
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
```

Vous pouvez utiliser les fichiers audio non seulement à partir d'un dossier **le répertoire_du terminal\ Sounds** mais aussi de n'importe quel sous-dossier qui se trouve dans le dossier **le répertoire_des données_du terminal\ MQL5.** On peut savoir la disposition du répertoire des données du terminal sur l'ordinateur dans le menu du terminal "Fichier" - "Ouvrir le répertoire des données" ou par la voie programmatique:

```
//--- Le dossier, où se trouvent les données du terminal
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
```

Par exemple, si un fichier audio Demo.wav se trouve dans le dossier le répertoire_des données_du terminal\ MQL5\Files, l'appel du PlaySound () doit être écrit ainsi:

```
//--- jouons un fichier audio Demo.wav à partir du dossier le répertoire_des données_d
   PlaySound("\\Files\\Demo.wav");
```

© 2000-2015, MetaQuotes Software Corp.

Prêtez l'attention que le chemin vers le fichier est écrite avec l'utilisation du symbole "\" dans le commentaire, et on utilise l'héritage "\\" pour diviser les dossiers dans la fonction.

A l'indication du chemin utilisez toujours seulement la ligne double inverse oblique comme séparateur, puisque la ligne séparée inverse est le symbole dirigeant pour le compilateur à l'analyse des lignes constantes et des constantes symboliques dans le code initial du programme.

Для остановки воспроизведения файла нужно вызвать функцию PlaySound() с параметром NULL:

```
//--- вызов PlaySound() с параметром NULL останавливает воспроизведение звука
   PlaySound(NULL);
```

ObjectCreate()

Un exemple d'un expert qui crée l'objet "Marque graphique" (OBJ_BITMAP_LABEL) à l'aide de la fonction ObjectCreate().

```
string label_name="currency_label";        // le nom de l'objet OBJ_BITMAP_LABEL
string euro      ="\\Images\\euro.bmp";    // le chemin vers le fichier le répertoire
string dollar    ="\\Images\\dollar.bmp";  // le chemin vers le fichier le répertoire
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- créons le bouton OBJ_BITMAP_LABEL, s'il est encore absent
   if(ObjectFind(0,label_name)<0)
     {
      //--- essayons de créer l'objet OBJ_BITMAP_LABEL
      bool created=ObjectCreate(0,label_name,OBJ_BITMAP_LABEL,0,0,0);
      if(created)
        {
         //--- attacherons le bouton à l'angle droit supérieur du graphique
         ObjectSetInteger(0,label_name,OBJPROP_CORNER,CORNER_RIGHT_UPPER);
 //--- maintenant configurons les propriétés d'un objet
         ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,100);
         ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,50);
         //--- oblitérons le code de la dernière erreur au 0
         ResetLastError();
         //--- chargerons l'image pour l'état du bouton "Est appuyé"
         bool set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,0,euro);
         //--- vérifions le résultat
         if(!set)
           {
            PrintFormat("On n' a pas réussi à charger l'image du fichier %s. Le code
           }
         ResetLastError();
         //--- chargerons l'image pour l'état du bouton "Est pressé"
```

© 2000-2015, MetaQuotes Software Corp.

```
         set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,1,dollar);

         if(!set)
           {
            PrintFormat("On n' a pas réussi à charger l'image du fichier %s. Le code d
           }
         //--- rendons la commande de la mise à jour au graphique pour que le bouton a
         ChartRedraw(0);
        }
      else
        {
         //--- on n'a pas réussi à créer l'objet, annonçons cette information
         PrintFormat("On n'a pas réussi à créer l'objet OBJ_BITMAP_LABEL. Le code de l
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprimons l'objet du graphique
   ObjectDelete(0,label_name);
  }
```

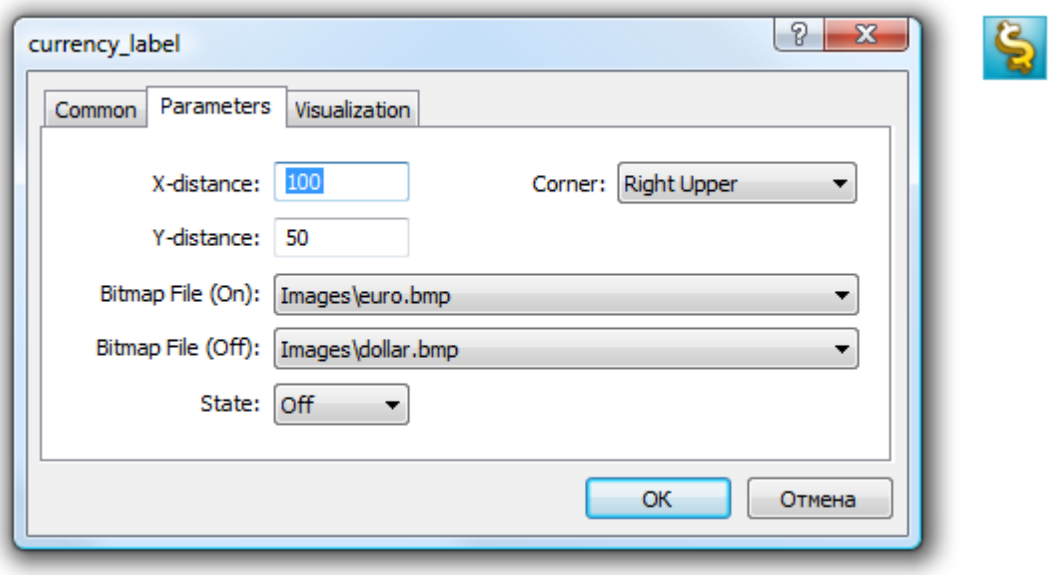
La création et la configuration d'un objet graphique avec le nom currency_label se passe dansla fonction OnInit (). Les chemins vers les fichiers chargés graphiques sont spécifiés dans les variables globales **euro** et **dollar**, comme un séparateur est utilisée une double barre oblique inversée:

```
string euro      ="\\Images\\euro.bmp";    // le chemin vers le fichier le répertoire_
string dollar    ="\\Images\\dollar.bmp";  // le chemin vers le fichier le répertoire_
```

Les fichiers eux-mêmes se trouvent dans ce dossier **le répertoire_des données_du terminal**\MQL5 \Images.

L'objet OBJ_BITMAP_LABEL représente le bouton, qui en fonction de l'état (est appuyé ou pressé) peut afficher une de deux images :euro.bmp ou dollar.bmp.

© 2000-2015, MetaQuotes Software Corp.



Les dimensions du bouton avec l'interface graphique s'établissent automatiquement à la dimension de l'image affichée. Le changement de l'image est produit à la pression par le bouton gauche de la souris sur l'objet OBJ_BITMAP_LABEL (**dans les propriétés il faut choisir "Désactiver la sélection"**). L'objet OBJ_BITMAP est créé de la même manière et est destiné à la création du fond avec le dessin demandé.

On peut changer dynamiquementla valeur de la propriétéOBJPROP_BMPFILE, qui est responsable de l'apparence des objets OBJ_BITMAP et OBJ_BITMAP_LABEL . Cela permet de créer de diverses interfaces interactives d'utilisateur pour les programmes mql5.

L'insertion des ressources aux fichiers exécutés à la compilation des programmes mql5

Pour le travail du programme mql5 peut être nécessaire la multitude de diverses ressources chargées en forme des fichiers des images et des sons. Pour exclure la nécessité du transfert de tous ces fichiers à la transmission du programme exécuté sur MQL5, il faut utiliser la directive du compilateur #resource:

```
#resource le chemin_vers_le fichier_de la ressource
```

La commande #resourceindique au compilateur que la ressource selon un chemin spécifié **le chemin_vers_le fichier_de la ressource** doit être inclus dans le fichier exécuté EX5. Ainsi, toutes les images et les sons peuvent être placés directement dans le fichier EX5 et pour exécuter le programme dans un autre terminal on n'a pas besoin d'envoyer tous les fichiers séparés qui sont y utilisés. N'importe quel fichier EX5 peut contenir les ressources et n'importe quel programme EX5 peut utiliser les ressources d'un autre programme EX5.

Les fichiers en format BMP et WAV sont automatiquement compressées avant l'inclusion dans un fichier exécutable EX5. Cela signifie que l'utilisation des ressources permet de créer les programmes complets sur MQL5, ainsi que réduire également la taille des fichiers qui sont nécessaire au terminal à l'utilisation du graphique et du son par rapport à la façon traditionnelle de l'écriture des programmes MQL5.

© 2000-2015, MetaQuotes Software Corp.

Le montant du fichier de la ressource ne peut pas être plus de 16 Mb.

La recherche des ressources indiquées par le compilateur

La ressource est inséré par la commande # ressource "<le chemin vers le fichier de la ressource>"

```
#resource "<le chemin_vers_le fichier_de la ressource>"
```

La longueur de la chaîne constante <le chemin_vers_le fichier_de la ressource> ne doit pas dépasser 63 caractères.

Le compilateur cherche la ressource selon le chemin indiqué dans l'héritage suivant:

- si au début du chemin il y a le séparateur la barre oblique inversée "\" (est écrit "\\"), la ressource est recherchée par rapport au répertoire **le répertoire_des données_du terminal**\MQL5\,
- s'il n'y a pas la barre oblique inversée, la ressource est recherchée par rapport la disposition du fichier initial dans lequel cette ressource est prescrit.

Il est inadmissible d'utiliser les sous - chaînes "..\\" et ":\\" dans le chemin de la ressource.

Les exemples de l'insertion des ressources:

```
//--- l'indication juste des ressources
#resource "\\Images\\euro.bmp" // euro.bmp se trouve dans le répertoire_des données_du
#resource "picture.bmp"        // picture.bmp se trouve dans le même répertoire que le
#resource "Resource\\map.bmp"  // la ressource se trouve dans le dossier le répertoire

//--- l'indication incorrecte des ressources
#resource ":picture_2.bmp"     // impossible d'utiliser ":"
#resource "..\\picture_3.bmp"  // impossible d'utiliser ".."
#resource "\\Files\\Images\\Folder_First\\My_panel\\Labels\\too_long_path.bmp" //plus
```

Utilisation des ressources

Nom de la ressource

Quand la ressource est déclarée comme la directive #resource, on peut l'utiliser dans n'importe quelle partie du programme. Le nom de la ressource devient son chemin sans la ligne oblique au début de la ligne qui spécifie le chemin vers la ressource. Pour utiliser sa ressource dans le code il faut ajouter un signe spéciale "::" avant le nom de la ressource.

Exemples:

```
//--- les exemples de l'indication des ressources et leurs noms au commentaire
#resource "\\Images\\euro.bmp"          // le nom de la ressource - Images\euro.bmp
#resource "picture.bmp"                 // le nom de la ressource - picture.bmp
#resource "Resource\\map.bmp"           // le nom de la ressource - Resource\map.bmp
#resource "\\Files\\Pictures\\good.bmp" // le nom de la ressource - Files\Pictures\goo
#resource "\\Files\\Demo.wav";          // le nom de la ressource - Files\Demo.wav"
#resource "\\Sounds\\thrill.wav";       // le nom de la ressource - Sounds\thrill.wav'
```

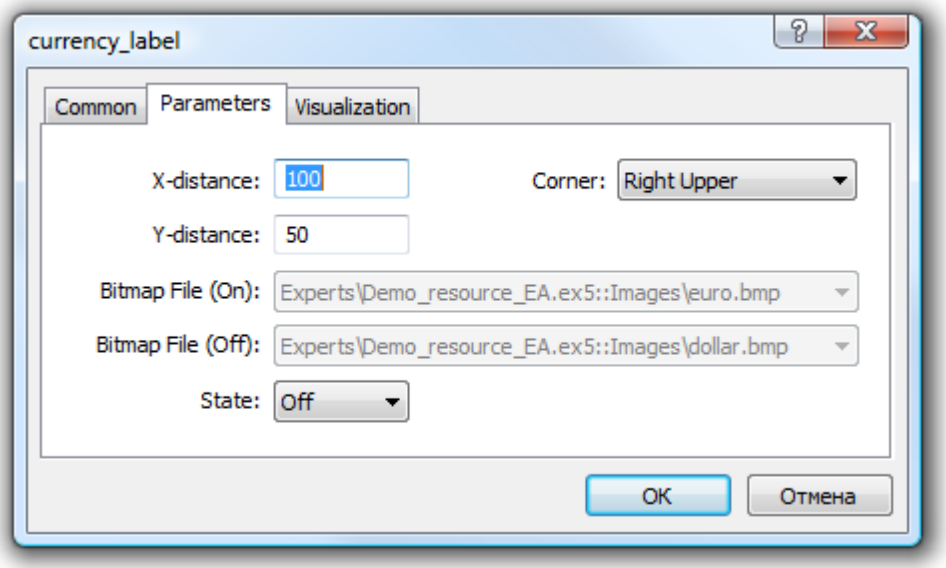
© 2000-2015, MetaQuotes Software Corp.

```
...

//--- utilisation des ressources
ObjectSetString(0,bitmap_name,OBJPROP_BMPFILE,0,"::Images\\euro.bmp");
...
ObjectSetString(0,my_bitmap,OBJPROP_BMPFILE,0,"::picture.bmp");
...
set=ObjectSetString(0,bitmap_label,OBJPROP_BMPFILE,1,"::Files\\Pictures\\good.bmp");
...
PlaySound("::Files\\Demo.wav");
...
PlaySound("::Sounds\\thrill.wav");
```

Il faut noter que pendant l'installation de l'image de la ressource aux objets OBJ_BITMAP et OBJ_BITMAP_LABEL, on ne peut pas changer manuellement la valeur de la propriété OBJPROP_BMPFILE. Par exemple, supposons que nous utilisons les ressources des fichiers euro.bmp et dollar.bmp. pour créer OBJ_BITMAP_LABEL.

```
#resource "\\Images\\euro.bmp";    // euro.bmp se trouve dans le répertoire_des donnée
#resource "\\Images\\dollar.bmp";  // dollar.bmp se trouve dans le répertoire_des donn
```

Alors à la vue des propriétés de cet objet nous verrons que les propriétés BitMap File (On) et BitMap File (Off) ont la couleur grise et sont inaccessibles pour le changement à la main:



Utiliser les ressources d'autres programmes mql5

L'utilisation des ressources a un autre avantage - on peut utiliser les ressources de chaque fichier EX5 dans chaque programme mql5. Ainsi, on peut utiliser les ressources d'un fichier EX5 dans les autres programmes mql5.

Pour utiliser le nom de la ressource d'un fichier tiers, il faut le spécifier comme <le chemin_du nom_du fichier_EX5>::<le nom_de la ressource>. Par exemple, si la ressource pour une image dans le fichier

© 2000-2015, MetaQuotes Software Corp.

triangle.bmp est spécifiée dans le script Draw_Triangles_Script.mq5:

```
#resource "\\Files\\triangle.bmp"
```

Puis son nom pour l'utilisation dans le script soit comme "Files\triangle.bmp", et pour l'utilisation il faut ajouter un signe spécial "::" pour le nom de la ressource.

```
//--- utilisation de la ressource dans le script
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"::Files\\triangle.bmp");
```

Pour utiliser la même ressource d'un autre programme, par exemple comme un expert, il faut ajouter supplémentairement au nom de la ressource le chemin vers le fichier EX5 rélativement au dossier **le répertoire_des données_du terminal\ MQL5** et le nom EX5 du fichier de ce script - **Draw_Triangles_Script.ex5** . Si le script se trouve dans un dossier standard**le répertoire_des données_du terminal\MQL5**Scripts\, il faut écrire l'appel de telle façon:

```
//--- utiliser la ressource du script dans l'expert
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"\\Scripts\\Draw_Triangles_Script.e
```

Si ne pas spécifier le chemin vers le fichier exécuté à l'appel à la ressource dans un autre fichier EX5, le fichier exécuté est recherché dans le même dossier, où se trouve le programme adressé de la ressource. Cela signifie que si dans le conseiller est demandée la ressource à partir du fichier Draw_Triangles_Script.ex5 sans chemin, par exemple:

```
//--- demande d'unu ressource du script dans l'expert sans chemin
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"Draw_Triangles_Script.ex5::Files\\
```

ainsi le fichier sera recherché dans le dossier **le répertoire_des données_du terminal\MQL5** Experts \, si le conseiller se trouve dans le dossier **le répertoire_des données_du terminal\MQL5** Experts\.

Le travail avec les indicateurs d'utilisateur connectés à titre des ressources

Pour le travail des programmes mql5 peuvent être nécessaires un ou plusieurs indicateurs d'utilisateur, ils peuvent être insérés dans le code du programme mql5 exécuté. L'insertion des indicateurs à titre des ressources permet de simplifier la diffusion des programmes.

L'exemple de la connexion et l'utilisation de l'indicateur d'utilisateur SampleIndicator.ex5 disposé dans le dossier: **le répertoire_des données_du terminal**\MQL5\Indicators\:

```
//+------------------------------------------------------------------+
//|                                                     SampleEA.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#resource "\\Indicators\\SampleIndicator.ex5"
int handle_ind;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//---
   handle_ind=iCustom(_Symbol,_Period,"::Indicators\\SampleIndicator.ex5");
   if(handle_ind==INVALID_HANDLE)
     {
      Print("Expert: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
     }
//--- ...
   return(INIT_SUCCEEDED);
  }
```

Le cas, quand l'indicateur d'utilisateur dans la fonction OnInit() crée une ou quelques copies de soi-même, demande la considération séparée. Rappellerons que pour l'utilisation de la ressource du programme mql5 il est nécessaire de l'indiquer comme: **<la voie_du nom_du fichier_EX5>::<le nom_de la ressource>**.

Par exemple, si l'indicateur SampleIndicator.ex5 fait partie du conseiller SampleEA.ex5 à titre de la ressource, la voie chez elle-même, indiqué à l'appel iCustom() en fonction de l'initialisation de l'indicateur d'utilisateur, aura l'aspect comme: "\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5". À l'indication évidente de cette voie l'indicateur d'utilisateur SampleIndicator.ex5 sera durement attaché au conseiller SampleEA.ex5 et perd la capacité du travail indépendant.

On peut recevoir la voie jusqu'à elle-même à l'aide de la fonction GetRelativeProgramPath (), dont l'exemple de l'utilisation est amené plus bas:

```
//+------------------------------------------------------------------+
//|                                                SampleIndicator.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property indicator_separate_window
#property indicator_plots 0
int handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le moyen incorrect de l'indication du lien a lui-mкme
//--- string path="\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5";
//--- le moyen correct de l'indication du lien a lui-mкme
  string path=GetRelativeProgramPath();
//--- indicator buffers mapping
   handle=iCustom(_Symbol,_Period,path,0,0);
   if(handle==INVALID_HANDLE)
     {
      Print("Indicator: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   else Print("Indicator handle=",handle);
//---
   return(INIT_SUCCEEDED);
  }
///....
//+------------------------------------------------------------------+
//| GetRelativeProgramPath                                           |
//+------------------------------------------------------------------+
string GetRelativeProgramPath()
  {
   int pos2;
//--- recevons la voie absolue vers le programme
   string path=MQLInfoString(MQL_PROGRAM_PATH);
//--- trouvons la position de la sous-chaone "\MQL5\"
   int    pos =StringFind(path,"\\MQL5\\");
//--- la sous-chaone n'est pas trouvйe - l'erreur
   if(pos<0)
      return(NULL);
//--- manquons le rйpertoire "\MQL5"
   pos+=5;
//--- manquons '\' superflus
   while(StringGetCharacter(path,pos+1)=='\\')
      pos++;
//--- si c'est la ressource, rendons la voie relativement le rйpertoire MQL5
   if(StringFind(path,"::",pos)>=0)
      return(StringSubstr(path,pos));
//---trouverons le sйparateur pour le premier sous-rйpertoire dans MQL5 (par exemple,
//--- s'il est absent, rendrons la voie relativement le rйpertoire MQL5
   if((pos2=StringFind(path,"\\",pos+1))<0)
      return(StringSubstr(path,pos));
//--- rendrons la voie relativement le rйpertoire (par exemple, MQL5\Indicators)
   return(StringSubstr(path,pos2+1));
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double& price[])
  {
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

ResourceCreate(), ResourceSave(), PlaySound(), ObjectSetInteger(), ChartApplyTemplate(), Opérations de fichier

© 2000-2015, MetaQuotes Software Corp.

L'appel des fonctions importées

Pour importer des fonctions pendant l'exécution du programme mql5 on utilise un enchaînement . Cela signifie que si dans le programme il y a un appel de la fonction importée, le module correspondant (ex5 ou dll) est chargé en train du chargement du programme. Les bibliothèques MQL5 et DLL sont accomplies dans le thread du module appelant.

Il n'est pas recommandé d'utiliser le nom entièrement qualifié du module chargé comme *Drive:\Directory\FileName.Ext*. Les bibliothèques MQL5 sont chargées du classeur *terminal_dir\MQL5\Libraries*. Si la bibliothèque n'était pas trouvée, on produit la tentative de charger la bibliothèque du classeur *terminal_dir\experts*.

Les bibliothèques systémiques (DLL) sont chargées selon les règles du système opérationnel. Si la bibliothèque est déjà chargée (par exemple, d'un autre expert et même d'un autre terminal de client lancé parallèlement), l'appel s'adresse à la bibliothèque déjà chargée. Dans le cas contraire la recherche va dans l'héritage suivant:

1. Le répertoire d'où on lançait le module, important dll. Sous le module on considère l'expert, le script, l'indicateur ou la bibliothèque EX5;
2. Le répertoire le répertoire_du_terminal_données\MQL5\Libraries (TERMINAL_DATA_PATH\MQL5\Libraries);
3. Le répertoire, d'où on lançait le terminal de client MetaTrader 5;
4. Le répertoire systémique;
5. Le répertoire Windows;
6. Le répertoire courant;
7. Les répertoires, énumérés dans une variable de système de l'entourage PATH.

Si la bibliothèque DLL utilise dans le travail l'autre DLL, en cas de l'absence de la deuxième DLL, la première ne pourra pas être chargé.

Avant le chargement de l'expert (le script, l'indicateur) on forme le répertoire total de toutes les modules de bibliothèque EX5, qu'on suppose d'utiliser comme de l'expert récupéré (le scénario, l'indicateur), et des bibliothèques de cette liste. On assure ainsi le chargement unique des modules plusieurs fois utilisées de bibliothèque EX5. Les bibliothèques se servent des variables prédéterminées de l'expert qui les a provoqué (le script, l'indicateur).

La recherche de la bibliothèque importée EX5 est produite dans la séquence suivante:

1. Le répertoire, le chemin vers lequel est donné par rapport au répertoire important EX5 de l'expert (le script, l'indicateur);
2. Le répertoire le répertoire_du terminal\MQL5\Libraries;
3. Le répertoire MQL5\Libraries dans le répertoire commun de tous les terminaux de client MetaTrader 5(Common\MQL5\Libraries).

Les fonctions importées de DLL au programme mql5, doivent assurer l'accord sur les liens, accepté pour les fonctions Windows API. Pour la garantie d'un tel accord dans le texte initial des programmes écrits dans les langages C ou C ++ on utilise le mot-clé __stdcall, qui est spécifique pour les compilateurs de la compagnie Microsoft(r). L'accord est caractérisé par:

- la fonction appelante (dans notre cas c'est le programme mq5) doit "voir" le prototype de la fonction appelée (importée de DLL), pour mettre correctement les paramètres à la pile;

© 2000-2015, MetaQuotes Software Corp.

- la fonction appelante (dans notre cas c'est le programme mq5) met les paramètres sur la pile dans l'ordre inverse, de droite à gauche - notamment en tel ordre la fonction importée lit les paramètres transmis à elle;
- les paramètres sont transmis selon la valeur, à l'exception de ceux qui sont transmis évidemment selon le lien (dans notre cas des lignes)
- la fonction importée, en lisant les paramètres transmis à elle, nettoie la pile.

A la description du prototype de la fonction importée on peut utiliser les paramètres avec les valeurs par défaut.

En cas si la bibliothèque correspondante n'a pas pu être chargée, ou on établit l'interdiction à l'utilisation DLL, ou la fonction importée n'était pas trouvée – l'expert arrête le travail avec le message correspondant "expert stopped" au journal. Ainsi l'expert ne fonctionnera pas avant que on ne le réinitialise pas. L'expert peut être réinitialisé à la suite de la recompilation ou après l'ouverture du tableau des propriétés de l'expert et la pression du bouton OK.

La transmission des paramètres.

Tous les paramètres des types simples sont transmis selon la valeur, si n'est pas indiqué évidemment qu'ils sont transmis selon le lien. A la transmission de la ligne on transmet l'adresse du tampon de la ligne copiée; si la ligne est transmise selon le lien, à la fonction importée de DLL, on transmet l'adresse du tampon notamment de cette ligne sans copiage.

Les structures, contenant les tableaux dynamiques, les lignes, les classes, d'autres structures complexes, ainsi que les tableaux dynamiques ou statiques des objets énumérés, ne peuvent pas être transmises à titre du paramètre à la fonction importée.

A la transmission à DLL du tableau toujours (indépendamment du drapeau AS_SERIES) on transmet l'adresse du début du tampon des données. La fonction au-dedans de DLL ne sait rien du drapeau AS_SERIES, le tableau transmis est le tableau statique de la longueur inconnue, pour l'indication de la grandeur du tableau il faut utiliser une paramétrisation supplémentaire.

© 2000-2015, MetaQuotes Software Corp.

Les erreurs de l'exécution

Dans le sous-système exécutif du terminal de client il y a une possibilité de sauvegarde du code de l'erreur dans le cas de son occurrence à l'exécution du programme mql5. Pour chaque programme mql5 exécuté on prévoit une variable prédéterminée _LastError.

Avant le lancement la fonction OnInit la variable _LastError remet au zéro. A l'apparition de la situation fausse pendant les calculs ou en train de l'appel de la fonction insérée la variable _LastError prend le code correspondant de l'erreur. On peut recevoir la valeur sauvegerdé dans cette variable à l'aide de la fonction GetLastError().

Il y a une série d'erreurs critiques, à l'apparition de lesquelles l'exécution du programme s'interrompt immédiatement:

- la division en le zéro;
- la sortie en dehors du tableau;
- l'utilisation de l'indicateur sur l'objet incorrect;

© 2000-2015, MetaQuotes Software Corp.

Le test des stratégies commerciales

L'idée du commerce automatique est attrayante par ce que le robot commercial peut travailler 24 heures par jour et sept jours par semaine. Le robot ne sait pas la fatigue, le doute et la peur, les problèmes psychologiques. Il est assez formaliser nettement les règles commerciales et les réaliser en forme des algorithmes, et le robot est prêt à travailler infatigablement. Mais autrefois il est nécessaire de se persuader de l'observance de deux importantes conditions:

- l'expert fait les opérations commerciales conformément aux règles du système commercial;
- la stratégie commerciale réalisée dans l'expert, montre le bénéfice sur l'histoire.

Pour recevoir des réponses à ces questions il y a le testeur de stratégie, faisant partie du terminal de client MetaTrader 5.

Dans ce paragraphe on examine toutes les particularités du test et l'optimisation des programmes dans le testeur des stratégies:

- Ограничения работы функций в тестере торговых стратегий
- Les modes de la génération des ticks
- La modélisation du spread
- Les variables globales du terminal de client
- Le calcul des indicateurs au test
- Le chargement de l'histoire au test
- Le test multidevise
- La modélisation du temps dans le testeur
- Les objets graphiques au test
- La fonction OnTimer() dans le testeur
- La fonction Sleep() dans le testeur
- L'utilisation du testeur pour les tâches de l'optimisation dans les calculs mathématiques
- La synchronisation des barres au test en mode "Seulement les prix de l'ouverture"
- La fonction IndicatorRelease() dans le testeur
- Le traitement des événements dans le testeur
- Les agents du test
- L'échange des données entre le terminal et l'agent
- L'utilisation du dossier commun de tous les terminaux de client
- L'utilisation de DLL

Ограничения работы функций в тестере торговых стратегий

Существуют ограничения работы некоторых функций в тестере стратегий клиентского терминала.

Функции Print() и PrintFormat()

© 2000-2015, MetaQuotes Software Corp.

Для увеличения быстродействия при оптимизации параметров советника функции Print() и PrintFormat() не выполняются. Исключением является использование этих функций внутри обработчика OnInit(). Это позволяет облегчить поиск причин ошибок при их возникновении.

Функции Alert(), MessageBox(), PlaySound(), SendFTP, SendMail(), SendNotification(), WebRequest()

Функции взаимодействия с "внешним миром" Alert(), MessageBox(), PlaySound(), SendFTP(), SendMail(), SendNotification() и WebRequest() в тестере стратегий не выполняются.

Les modes de la génération des ticks

L'expert en langue MQL5 c'est le programme, qui s'exécute chaque fois en réponse à une certaine influence extérieure - l'événement. Pour chaque événement prédéterminé dans l'expert il y a une fonction correspondant à cet événement - le gestionnaire de l'événement.

L'événement principal pour l'expert c'est le changement de prix - NewTick, et c'est pour cela pour le test des experts il est nécessaire de générer les successions de tick. Dans le testeur du terminal de client MetaTrader 5 on a réalisé 3 modes de la génération des ticks:

- Tous les ticks
- Les prix OHLC des barres momentanées (1 Minute OHLC)
- Seulement les prix de l'ouverture

Le mode de base et celui le plus détaillé de la génération c'est le mode "Tous les ticks", les autres deux modes sont la simplification de l'essentiel et seront décrits par comparaison avec le mode "Tous les ticks". Examinerons tous les trois modes pour comprendre quelle est la différence entre eux.

Tous les ticks

L'histoire des cotations selon les instruments financiers est transmise du serveur commercial au terminal de client MetaTrader 5 en forme des blocs économiquement emballés des barres momentanées. L'information détaillée comment se passe la demande et la construction des temps trames necéssaire on peut lire dans le paragrapheOrganisation de l'accès aux données.

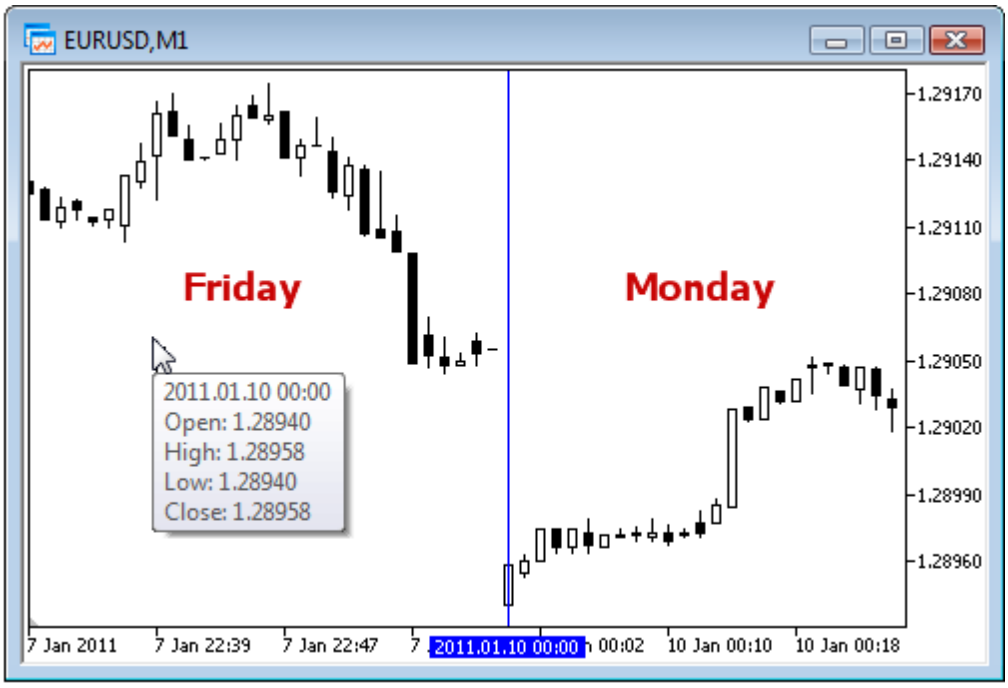
L'élément minimal de l'histoire de prix est la barre momentanée, d'où on peut recevoir l'information sur quatre vakeurs du prix:

- Open - le prix, selon lequel la barre momentanée est ouverte;
- High - le maximum, qui était atteint pendant cette barre momentanée;
- Low - le minimum, qui était atteint pendant cette barre momentanée;
- Close - le prix de la clôture du bar.

Une nouvelle barre momentanée s'ouvre non au moment où commence une nouvelle minute (la quantité de secondes devient égale à 0), mais quand le tick vient- le changement du prix quand même sur un point. Sur le dessin est montré la première barre momentanée de la nouvelle semaine commerciale, qui a le temps de l'ouverture 2011.01.10. 00:00. La rupture de prix entre vendredi et lundi, lequel nous voyons sur le graphique, est le fait ordinaire, puisque même aux jours fériés les

© 2000-2015, MetaQuotes Software Corp.

cours du change changent en réponse aux nouvelles entrantes.

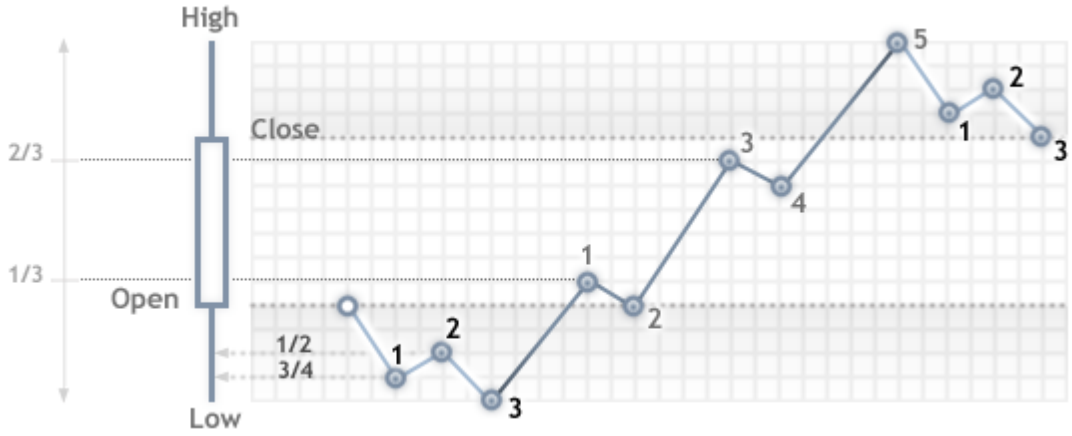


Pour cette barre nous savons seulement que la barre donnée momentanée est ouverte le 10 janvier 2011 à 00 heures 00 minutes, mais on ne sait rien sur les secondes. Ce pouvait être le temps 00:00:12 ou 00:00:36 (12 ou 36 secondes après le début du nouveau jour) ou n'importe quel autre temps dans la limite de cette minute. Mais nous connaissons exactement qu'au moment de l'ouverture d'une nouvelle barre momentanée le prix Open selon EURUSD était au niveau 1.28940.

De la même manière nous ne connaissons pas à près la seconde, quand le tick correspondant au prix de la clôture de la barre examinée momentanée est venu, on sait seulement une chose - c'est un dernier prix de la barre momentanée, qui était inscrite comme le prix Close. Pour l'instant, c'était le prix1.28958. Le temps de l'apparition des prix High et Low est inconnu aussi, mais nous savons que les prix maximum et minimum ont été aux niveaux 1.28958 et 1.28940 conformément.

Pour tester la stratégie commerciale, nous avons besoin de la succession de tick, auquelle sera émulé le travail d'un expert. Ainsi, pour chaque barre momentanée nous savons **4 points de contrôle**, où le prix a été. Si la barre a seulement 4 tics, c'est suffisant pour le test de cette information, mais le volume de tick est plus 4 d'habitude. Donc il est nécessaire de générer les points supplémentaires de contrôle pour les ticks, qui venaient entre les prix Open, High, Low et Close. Le principe de la génération des ticks en mode "Tous les ticks" est décrit dans l'article Algorithme de la génération des ticks dans le testeur des stratégies du terminal MetaTrader 5, dont le dessin est présenté plus bas.

© 2000-2015, MetaQuotes Software Corp.



Au test en mode "Tous les ticks" la fonction OnTick()de l'expert sera **appelée à chaque point de contrôle**, chaque point de contrôle c'est un tick de la succession générée. L'expert recevra le temps et le prix du tick modelé de la meme façon qu'au travail en ligne.

Important:le mode du test "Tous les ticks" est le plus exact, mais en cela il prend beaucoup de temps. Pour l'évaluation initiale de la plupart des stratégies commerciales il suffit d'utiliser l'un de deux autres modes du test.

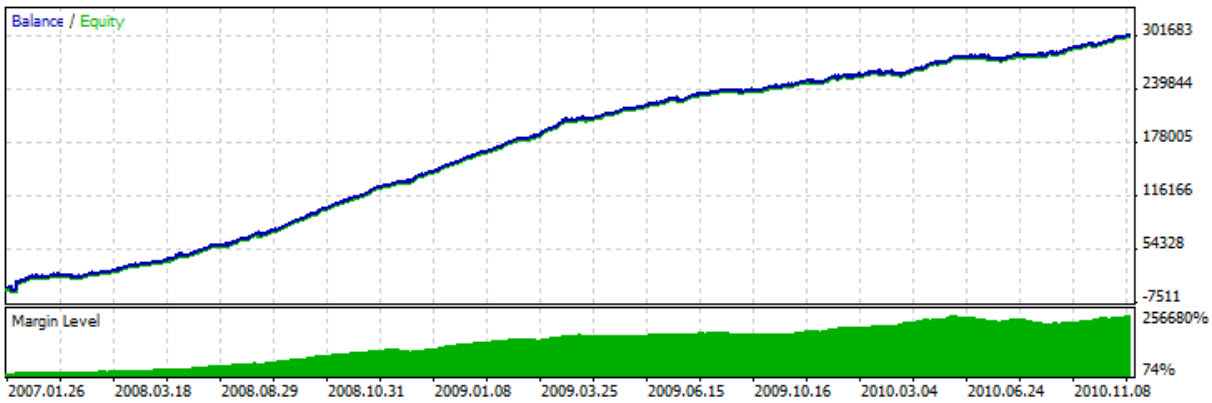
1 minute OHLC

Le test en mode "Tous les ticks" est le plus précis des trois modes, mais en même temps est le plus lent. Le lancement du gestionnaire OnTick() se passe sur chaque tick, et le volume de tick peut être assez grand. Pour les stratégies, pour lesquelles il n'est pas important, dans quelle succession de tick se développait le prix au cours de la barre, il y a un mode plus rapide et plus grossier de la modélisation - c'est "1 minute OHLC".

En mode "1 minute OHLC" la succession de tick est construite seulement selon**les prix OHLC des barres momentanées**, le nombre de points générés de contrôle diminue beaucoup - donc, diminue le temps du test. Le lancement de la fonction OnTick () est produite sur tous les points de contrôle, qui sont construits aux prix OHLC des barres momentanées.

Le refus de la génération des ticks supplémentaires intermédiaires entre les prix Open, High, Low et Close amène à l'apparition de la détermination rigide dans le développement du prix du moment, quand le prix Open est déterminé. Il est donc possible de créer "Graal de test", qui montre un beau graphique montant de la balance au test. L'exemple d'un tel Graal est présenté à Code Base - Grr-al.

© 2000-2015, MetaQuotes Software Corp.



Sur le dessin est présenté le graphique très attrayant du test de cet expert. Comment est-il reçu? 4 prix sont connus pour la barre momentanée et pour eux il est connu que le premier prix est Open, et le dernier est le prix Close. Entre eux il y a les prix High et Low, la succession de leur arrivée est inconnue, mais on sait que le prix High est plus ou est égal au prix Open (le prix Low est moins ou est égal au prix Open).

Il suffit de définir le moment de l'entrée du prix Open et puis analyser le tick suivant pour définir que devant nous sont High ou Low. Si le prix est plus bas que le prix Open, donc devant nous est le prix Low - achetons sur ce tick, le tick suivant correspondra au prix High, sur qui nous fermons l'achat et nous ouvrons la vente. Le tick suivant est le dernier, c'est le prix Close, nous y fermons la vente.

Si après le prix le tick avec le prix plus de prix de l'ouverture est venu, la succession des marchés est inverse. Traitons en tel mode la barre momentanée et attendons la suivant. Au test d'un tel expert tout va bien à l'histoire, mais quand l'exécute en ligne, et la situation change de façon spectaculaire - la ligne de la balance est toujours égal, mais descend. Pour une révélation rapide du truc il suffit de lancer un tel conseiller en mode "Tous les ticks".

Important:si les résultats du test de l'expert dans les modes grossiers du test ("1 minute OHLC" et "Seulement les prix de l'ouverture") sont trop bons, testez-les en mode "Tous les ticks".

Seulement les prix de l'ouverture

Dans ce mode, il y a la génération des ticks selon les prix OHLC du temps trame sélectionné pour le test. En cela la fonction de l'expert OnTick () est lancée seulement au début de la barre au prix Open. A cause de cette particularité les niveaux - stops et les ordres remis peuvent fonctionner non au prix déclaré (particulièrement au test des temps trames plus élevés). En échange de cela, nous sommes en mesure de faire rapidement le test estimatif d'un expert.

Par l'exception à la génération des ticks en mode "Seulement les prix de l'ouverture" sont les périodes W1 et MN1 : pour ces temps trames les ticks sont générés pour les prix OHLC de chaque jour, et non pour les prix OHLC de la semaine ou du mois respectivement.

Par exemple, on produit le test du conseiller sur EURUSD H1 en mode "Seulement les prix de l'ouverture". Dans ce cas le nombre total des ticks (les points de contrôle) sera pas plus 4* nombre de barres d'heures qui se trouvent dans l'intervalle testé. Mais en cela**l'appel du gestionnaire OnTick() est produit seulement sur l'ouverture** de la barre**d'heure** . Sur les autres ticks ("cachés" de l'expert) se produisent les vérifications nécessaires pour le test correct:

- le calcul des exigences de marge;

© 2000-2015, MetaQuotes Software Corp.

- le fonctionnement de Stop Loss et Take Profit;
- le fonctionnement des ordres remis;
- la suppression des ordres remis avec le temps passé.

S'il n'y a pas de positions ouvertes ou des ordres remis, il n'y a pas de nécessité de ces vérifications sur les ticks cachés et l'accroissement de la vitesse peut être important. Ce mode "Seulement les prix de l'ouverture" convient bien au test des stratégies, qui font les marchés seulement sur l'ouverture de la barre et n'utilisent pas les ordres remis, ainsi que n'utilisent pas les ordres StopLoss,. TakeProfit. Pour la classe de telles stratégies se sauvegarde toute l'exactitude nécessaire du test.

Spécifions le conseiller Moving Average de la livraison standard en qualité de l'exemple d'un expert, à qui n'est pas important dans quel mode passer le test. La logique de cet expert est construite de telle manière que toutes les décisions sont acceptées sur l'ouverture de la barre et les marchés sont passés tout de suite sans utilisation des ordres remis. Lancerons le test de l'expert sur EURUSD H1 sur l'intervalle du 2010.01.09 jusqu'au 2010.31.12 et comparerons les graphiques. Sur le dessin sont montrés les graphiques de la balance du rapport du testeur pour tous trois modes.

Comme vous le voyez, les graphiques sur les modes différents du test sont absolument identiques pour le conseiller Moving Average de la livraison standard.

Il y a une série de limitations de l'application du mode "Seulement les prix de l'ouverture":

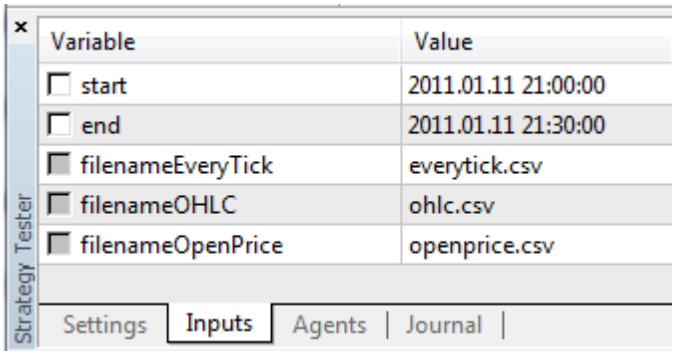
- On ne peut pas utiliserle mode du commerce "Interdiction hasardée";
- Dans l'expert testé il est impossible de s'adresser aux données dutemps trame plus bas qu'on utilise pour le test/optimisation. Par exemple, si le test/optimisation se réalise à la période H1, vous pouvez vous adresser aux données H2, H3, H4 etc., mais non aux données M30, M20, M10 etc. En dehors de cela, les temps trames plus grands auxquels il y a un appel, doivent être divisibles au temps trame du test. Par exemple, au test à la période M20 on ne peut pas s'adresser au temps trame M30, mais on peut s'adresser à H1. Ces limitations sont conditionnées par l'impossibilité de recevoir les données des temps trames plus bas et non divisible des barres générées pendant le test/optimisation.
- Les limitations selon l'appel aux données des autres temps trames se répandent sur les autres

© 2000-2015, MetaQuotes Software Corp.

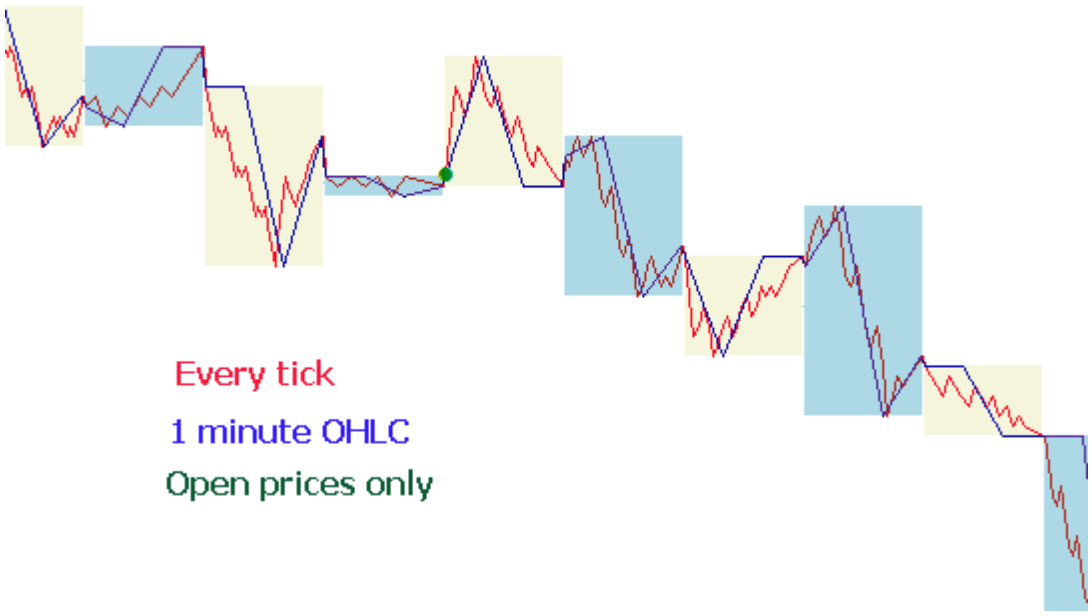
symboles, dont les données sont utilisés par le conseiller. Cependant, dans ce cas, la limitation pour chaque symbole est le premier temps trame à laquel l'appel a eu lieu pendant le test/optimisation. Par exemple, le test se réalise sur le symbole et la période EURUSD H1, le conseiller s'est adressé pour la première fois au symbole GBPUSD M20. Dans cette situation, le conseiller peut utiliser les données EURUSD H1, H2, etc à l'avenir, ainsi que GBPUSD M20, H1, H2 etc.

Important: le mode "Seulement les prix de l'ouverture" est le plus rapide par le temps du test, mais ne convient pas pour toutes les stratégies commerciales. Choisissez le mode nécessaire du test à partir des particularités du travail du système commercial.

A l'achèvement du paragraphe sur les modes de la modélisation amènerons la comparaison visuelle des modes différents de la génération des tics pour EURUSD pour deux barres M15 sur l'intervalle de 2011.01.11 21:00:00 - 2011.01.11 21:30:00. L'enregistrement des ticks est produite à l'aide de l'expert WriteTicksFromTester.mq5 aux fichiers différents, la fin des noms de ces fichiers est spécifiiée dans paramètres input filenamEveryTick, filenameOHLC et filenameOpenPrice.



Pour recevoir trois fichiers avec trois successions de tick (pour chacun des modes "Tous les tics", "OHLC sur les barres momentanées" et "Seulement les prix de l'ouverture") le conseiller était lancé trois fois en modes correspondants aux passages séparés. Puis les données de ces trois fichiers à l'aide de l'indicateur TicksFromTester.mq5 sont déduites sur le graphique. Le code de l'indicateur se joint à l'article.



© 2000-2015, MetaQuotes Software Corp.

Par défaut toutesles opérations de fichier dans la langue MQL5 sont produites dans les limites "du sandbox de fichier" et au test seulement "le sandbox de fichier" personnel est accessible à l'expert. Pour que l'indicateur et l'expert au test travaillent avec les fichiers d'un dossier, le drapeau FILE_COMMONétait utilisé. L'exemple du code de l'expert:

```
//--- ouvrons le fichier
   file=FileOpen(filename,FILE_WRITE|FILE_CSV|FILE_COMMON,";");
//--- vérifions le succès de l'opération
   if(file==INVALID_HANDLE)
     {
      PrintFormat("On n'a pas réussi à ouvrir le fichier pour l'enregistrement %s. Le
      return;
     }
   else
     {
      //---informons de l'enregistrement au dossier commun de tous les terminaux de cl
      PrintFormat("Le fichier sera enregistré dans le dossier %s",TerminalInfoString(T
     }
```

Dans l'indicateur pour la lecture des données on a aussi utiliséle drapeau FILE_COMMON, cela a permis d'éviter le transfert des fichiers nécessaires manuellement d'un dossier à l'autre.

```
//--- ouvrons le fichier
   int file=FileOpen(fname,FILE_READ|FILE_CSV|FILE_COMMON,";");
//---vérifions le succès de l'opération
   if(file==INVALID_HANDLE)
     {
      PrintFormat("On n'a pas réussi à ouvrir le fichier pour la lecture %s. Le code d
      return;
     }
   else
     {
      //--- informons de l'emplacement du dossier commun de tous les terminaux de clie
      PrintFormat("Le fichier sera lit du dossier %s",TerminalInfoString(TERMINAL_COMM
     }
```

La modélisation du spread

La différence entre les prix Bid et Ask s'appelle le spread. Le spread n'est pas modelé au test, il est pris à partir des données historiques. If the spread is less than or equal to zero in the historical data, then the last known (at the moment of generation) spread of is used by testing agent.

Le spread dans le testeur est toujours considéré comme flottant. C'est-à-direSymbolInfoInteger(symbol, SYMBOL_SPREAD_FLOAT) rend toujours true.

En outre dans les données historiques se trouvent les valeurs des volumes de tick et commerciaux. La structure spéciale est utilisée pour le stockage et la réception des données MqlRates:

© 2000-2015, MetaQuotes Software Corp.

```
struct MqlRates
  {
   datetime time;         // le temps de l'ouverture de la barre
   double   open;         // le prix de l'ouverture Open
   double   high;         // le prix le plus haut High
   double   low;          // le prix le plus bas Low
   double   close;        // le prix de la clôture Close
   long     tick_volume;  // le volume de tick
   int      spread;       // le spread
   long     real_volume;  // le volume boursier
  };
```

Les variables globales du terminal de client

Au test les variables globales du terminal de client sont aussi émulé, mais ils ne sont aucunement liés avec les variables globales vrais du terminal, dont on peut voir dans le terminal selon le bouton F3. Cela signifie que toutes les opérations avec les variables globales du terminal au test sont produites en dehors du terminal de client (dans l'agent du test).

Le calcul des indicateurs au test

En mode du temps réel les valeurs des indicateurs sont calculés sur chaque tick. Dans le testeur est accepté le modèle économe du calcul des indicateurs - les indicateurs sont recalculés seulement directement avant que l'on lance l'expert à l'exécution. Cela signifie que la conversion des valeurs des indicateurs est produite avant l'appel des fonctions OnTick(), OnTrade() et OnTimer().

> Ce n'est pas grave, s'il y a ou il n'y a pas d'appel de l'indicateur dans un gestionnaire concret de l'événement, tous les indicateurs dont les gestionnaires ont été créés par la fonctioniCustom() ou IndicatorCreate(), seront forcément recalculés avant l'appel de la fonction-gestionnaire de l'événement.

Donc, au test en mode "Tous les ticks" le calcul des indicateurs se passe avant chaque appel OnTick(). Si la minuterie est activée dans l'expert à l'aide de la fonctionEventSetTimer(), les indicateurs seront recalculés avant chaque appel du gestionnaireOnTimer(). Donc, le temps du test peut augmenter plusieurs fois à l'utilisation dans l'expert de l'indicateur écrit par façon non optimale.

Le chargement de l'histoire au test

L'histoire selon l'instrument testé est synchronisée et chargée par le terminal du serveur commercial avant le lancement du procès du test. En cela pour la première fois le terminal télécharge à la fois toute l'histoire accessible selon l'instrument testé du serveur commercial pour ne pas y s'adresser après. Seulement le chargement des nouvelles données se passe après.

L'agent du test reçoit du terminal de client l'histoire selon l'instrument testé tout de suite après le lancement du test. Si en train du test on utilise les données selon d'autres instruments (par exemple, l'expert multi-devises), dans ce cas l'agent du test demande l'histoire necéssaire au terminal de client au premier appel. Si les données historiques se trouvent au terminal, ils sont transmis à la fois sur les agents du test. Si les données ne sont pas disponibles, le terminal les demandera et téléchargera à partir du serveur, et ensuite les passera aux agents du test.

L'appel aux instruments supplémentaires se passe aussi dans le cas quand on calcule le prix du cours cross aux opérations commerciales. Par exemple, au test de la stratégie sur EURCHF avec la devise du

© 2000-2015, MetaQuotes Software Corp.

dépôt en dollars US avant le traitement par la première opération commerciale l'agent du test demande au terminal de client l'histoire selon EURUSD et USDCHF, bien qu'à la stratégie il n'y a pas d'accès direct à ces instruments.

Avant de tester la stratégie multi-devises est recommandé de télécharger toutes les données historiques nécessaires pour le terminal de client. Cela permettra d'éviter des retards au test/ optimisation liés avec le chargement de données. On peut télécharger l'histoire par exemple, en ouvrant les graphiques correspondants et leur défilement vers le début de l'histoire. L'exemple du chargement forcé de l'histoire au terminal commercial est amené à la documentation MQL5 dans le paragraphe Organisation de l'accès aux données.

Les agents du test reçoivent aussi à son tour l'histoire du terminal dans l'aspect paqueté. Au test répété le chargement de l'histoire du terminal par le testeur ne se passe déjà pas, parce qu'il y a les données du lancement précédent du testeur.

- Le terminal charge l'histoire du serveur commercial seulement une fois au premier appel de l'agent au terminal pour recevoir l'histoire pour un symbole testé. L'histoire est chargée dans l'aspect paqueté pour réduire le trafic.
- Les ticks ne sont pas envoyés par le réseau, ils sont générés sur les agents de test.

Le test multidevise

Le testeur permet de faire la vérification sur l'histoire des stratégies vendant sur quelques instruments. Tels experts s'appellent conventionnellement multidevises, puisque initialement dans les plateformes précédents le test a été fait seulement pour un instrument. Dans le testeur du terminal MetaTrader 5 on peut modeler le commerce selon tous les instruments accessibles.

L'histoire selon les instruments utilisés est chargée automatiquement par le testeur du**terminal de client** (pas du serveur commercial!) au premier appel à l'outil donné.

L'agent du test charge seulement l'histoire manquant avec une petite réserve pour assurer les données nécessaires dans l'histoire pour le calcul des indicateurs au moment du début du test. Le volume minimal de l'histoire au chargement du serveur commercial pour les trames de temps D1 et moins fait un an. Ainsi, si on lance le test à l'intervalle 2010.11.01-2010.12.01 (le test à l'intervalle d'un mois) avec la période M15 (chaque barre est égal à 15 minutes), on demandera l'histoire selon l'instrument à partir du terminal pour l'année 2010. Pour les trames de temps Weekly sera demandée l'histoire aux 100 barres que fait environ deux ans (52 semaines par an). Pour le test sur une trame de temps de mois Monthly l'agent demandera l'histoire pour 8 ans (12 mois * 8 ans = 96 mois).

Si on ne réussit pas pour quelques raisons d'assurer le nombre nécessaire des barres avant le test, **la date de début sera automatiquement approchée** du passé vers le présent pour assurer une telle réserve.

Pendant le test "Aperçu du marché" est aussi émulé, à partir de lequel on peut recevoirl'information sur les instruments. Par défaut au début du test à "l'Aperçu du marché" du testeur il y a seulement un symbole - le symbole sur lequel le test est lancé. Tous les symboles nécessaires sont connectés automatiquement à "l'Aperçu du marché" du testeur (pas de terminal!) à l'appel à eux.

© 2000-2015, MetaQuotes Software Corp.

Avant le test de l'expert multi-devises il est nécessaire de choisir les instruments demandés pour le test dans "Aperçu du marché" du terminal et charger les données à la profondeur nécessaire. Au premier appel au symbole "étranger" on produit automatiquement la synchronisation selon ce symbole entre l'agent du test et le terminal de client. Le symbole "étranger" c'est le symbole se distinguant de celui, sur lequel le test a été lancé.

L'appel aux données du symbole étranger se passent dans les cas suivants:

- l'utilisationdes fonctions des indicateurs techniques et IndicatorCreate()sur la paire symbole/trame de temps;
- la demande vers "l'Aperçu du marché" (Market Watch) selon le symbole étranger:

1. SeriesInfoInteger
2. Barres
3. SymbolSelect
4. SymbolIsSynchronized
5. SymbolInfoDouble
6. SymbolInfoInteger
7. SymbolInfoString
8. SymbolInfoTick
9. SymbolInfoSessionQuote
10. SymbolInfoSessionTrade
11. MarketBookAdd
12. MarketBookGet

- la demande à la série temporelle selon la paire symbole/période par les fonctions:

1. CopyBuffer
2. CopyRates
3. CopyTime
4. CopyOpen
5. CopyHigh
6. CopyLow
7. CopyClose
8. CopyTickVolume
9. CopyRealVolume
10. CopySpread

Quand se passe un premier appel au symbole étranger, le procès du test s'arrête et se passe le téléchargement de l'histoire selon la paire symbole/période du terminal à l'agent du test. La génération de la succession de tick pour ce symbole est activée simultanément.

Pour chaque instrument on génère la succession propre de tick conformément au mode choisi de la génération de ticks. En outre on peut demander évidemment l'histoire pour les symboles nécessaires à l'aide de l'appel de la fonctionSymbolSelect() dans le gestionnaire OnInit() - le chargement de l'histoire sera produit tout de suite avant le test du conseiller.

© 2000-2015, MetaQuotes Software Corp.

Ainsi, pour faire le test multi-devise dans le terminal de client MetaTrader 5 il ne faut pas entreprendre aucuns efforts supplémentaires Il suffit d'ouvrir les graphiques des instruments correspondants dans le terminal de client. L'histoire selon les symboles nécessaires sera automatiquement chargée du serveur commercial à condition que ces données s'y trouvent.

La modélisation du temps dans le testeur

Au test le temps local TimeLocal() est toujours égal au temps de serveur TimeTradeServer(). A son tour, le temps de serveur est toujours égal au temps, correspondant au temps GMT - TimeGMT(). Ainsi, toutes ces fonctions au test donnent le même temps.

L'absence de la différence entre GMT,le temps local et le temps de serveur dans le testeur est faite consciemment pour cette raison que la connexion avec le serveur ne peut pas être toujours. Les résultats du test doivent être identiques, indépendamment de la présence du lien. L'information sur le temps de serveur n'est pas stockée localement, elle est prise du serveur.

Les objets graphiques au test

La construction des objets graphiques ne se réalise pas pendant le test/optimisation. Ainsi, à l'appel aux propriétés de l'objet créé pendant le test/optimisation l'expert recevra les valeurs nulles.

> Cette limitation ne se répand pas au test en mode visuel.

La fonction OnTimer() dans le testeur

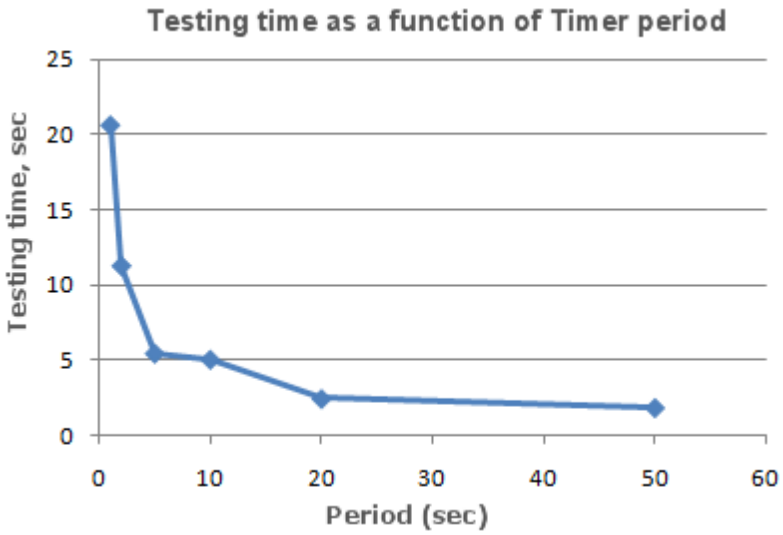
Le traitement des événements de la minuterie est possible à MQL5. L'appel du gestionnaireOnTimer() est produit indépendamment du mode du test. Cela signifie que si le test est lancé en mode "Seulement les prix de l'ouverture" sur la période H4 et à l'intérieur de l'expert est installée la minuterie avec l'appel chaque seconde, le gestionnaire OnTick () sera appelé une fois sur l'ouverture de chaque barre H4 et le gestionnaire OnTimer() sera appelé 14400 fois (3600 secondes * 4 heures) pendant la barre. Autant augmentera en cela le temps du test de l'expert, dépend de la logique de l'expert.

Pour vérifier la dépendance du temps du test de la périodicité spécifiée de la minuterie on a écrit l'expert simple sans opérations commerciales.

© 2000-2015, MetaQuotes Software Corp.

```
//--- input parameters
input int      timer=1;              // la signification de la minuterie, sec
input bool     timer_switch_on=true; // la minuterie est activée
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- lancerons la minuterie si  timer_switch_on==true
   if(timer_switch_on)
     {
      EventSetTimer(timer);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- arrêterons la minuterie
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
// ne faisons rien, le corps du gestionnaire est vide
  }
//+------------------------------------------------------------------+
```

Les mesures du temps du test ont été faites aux significations différentes du paramètre timer (la fréquence de l'événement Timer). Le graphique de la dépendance du temps du test T de la valur de la périodicité Period est construit sur les données reçues.



Il est clair, que plus petit est le paramètre timer à l'initialisation de la minuterie par la fonction EventSetTimer(timer), plus courte la période (Period) entre les appels du gestionnaire OnTimer(), et à

© 2000-2015, MetaQuotes Software Corp.

la suite le temps du test T aux mêmes les autres conditions est plus grand.

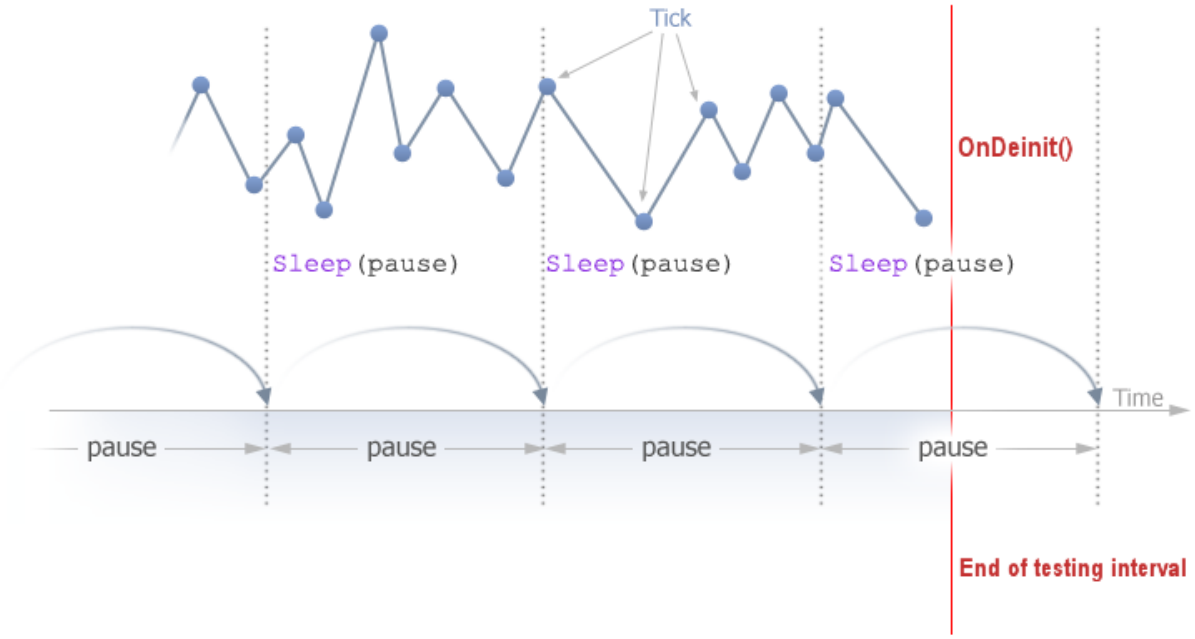
La fonction Sleep() dans le testeur

La fonctionSleep()permet arrêter l'exécution du programme mql5 dans l'expert ou dans le script pour un certain temps au travail sur le graphique. Cela peut être nécessaire à la demande de quelques données, qui ne sont pas encore prêts au moment de la demande et il est nécessaire d'attendre jusqu'à ce qu'ils soient prêts. On peut regarder l'exemple détaillé de l'utilisation de la fonction Sleep() dans le paragrapheOrganisation de l'accès aux données.

Dans le testeur les appels Sleep() ne retiennent pas le procès du test. A l'appel de Sleep() les ticks générés dans la limite du retard indiqué "sont joués", à la suite de quoi peuvent fonctionner les ordres remis, les stops etc.. Après l'appel de Sleep() le temps modelé dans le testeur s'augmente à l'intervalle indiquée dans le paramètre de la fonction Sleep.

Si à la suite de l'exécution de la fonction Sleep () le temps courant dans le testeur est sorti du cadre de la période du test, on reçoit l'erreur "la boucle infinie dans Sleep". A la réception d'une telle erreur les résultats du test ne sont pas rejetés, tous les calculs sont produits en totalité (la quantité de marchés, le prélèvement etc.) et les résultats du test donné sont transmis au terminal.

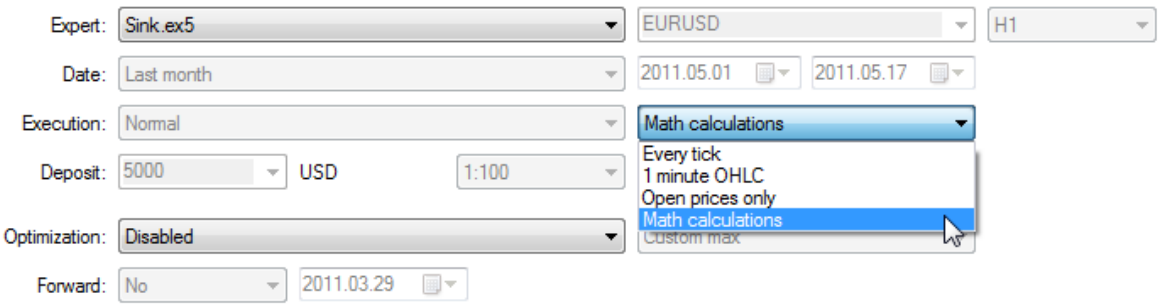
La fonction Sleep() ne fonctionne pas dans OnDeinit(), puisque après son appel le temps de test se trouve de manière garantie en dehors de l'intervalle du test.



L'utilisation du testeur pour les tâches de l'optimisation dans les calculs mathématiques

On peut utiliser le testeur dans le terminal MetaTrader 5 non seulement pour vérifier des stratégies commerciales, mais aussi pour les calculs mathématiques. Il est nécessaire de choisir pour cela le mode correspondant dans les réglages:

© 2000-2015, MetaQuotes Software Corp.



Au choix du mode "Calculs mathématiques" sera fait le passage "vide" de l'agent du test. Le passage vide signifie que la génération des ticks et le chargement de l'histoire ne seront pas faits. Seulement trois fonctions: OnInit(), OnTester(), OnDeinit() seront appelées à un tel passage.

Si la date de l'achèvement du test est moins ou est égale à la date de commencement du test, cela signifiera aussi les tests en mode "Calculs mathématiques".

A l'utilisation du testeur pour la décision des tâches mathématiques le chargement de l'histoire et la génération des ticks ne se passent pas.

Un problème typique mathématique pour la décision dans le testeur MetaTrader 5 - la recherche de l'extremum de la fonction de plusieurs variables. Il est nécessaire pour sa décision:

- Placer le bloc des calculs de la valeur de la fonction des plusieurs variables dans OnTester()et rendre la valeur calculée par return(la valeur_de la fonction);
- Porter les paramètres de la fonction au domaine global du programme dans l'aspect des variables input;

Compilons le conseiller, ouvrons la fenêtre "Testeur". Sur l'onglet "Paramètres d'entrée" marquons les variables demandées d'entrée et spécifions les frontières pour eux dans l'espace des valeurs et le pas pour le balayage.

Choisissons le type de l'optimisation - "Lent (le balayage complet des paramètres) ou "Rapide (un algorithme génétique)". Pour la recherche simple de l'extremum de la fonction il vaut mieux choisir l'optimisation rapide, mais s'il faut calculer les valeurs sur tout l'espace des variables, l'optimisation lente conviendra.

Choisissons le mode "Calculs mathématiques" et lançons la procédure de l'optimisation par le bouton "Démarrer". Il est nécessaire de se rappeler qu'à l'optimisation on cherche toujours le maximum local de la valeur de la fonction OnTester. Pour la recherche du minimum local on peut rendre de la fonction OnTester la valeur inverse à la valeur calculée de la fonction:

```
return(1/la valeur_de la fonction);
```

Il est nécessaire de réaliser indépendamment la vérification pour que la valeur_de la fonction n'est pas égal au zéro, autrement on peut recevoir l'erreur critiquede la division par zéro. Il y a une autre variante, plus convenant et ne déformant pas les résultats de l'optimisation, il est proposé par les lecteurs de l'article:

```
return(-la valeur_de la fonction);
```

Dans cette variante il ne faut pas vérifier la valeur_de la fonction à l'égalité au zéro et la surface elle-même des résultats de l'optimisation dans la représentation 3D a la même forme, mais seulement reflété symétriquement de l'initial.

© 2000-2015, MetaQuotes Software Corp.

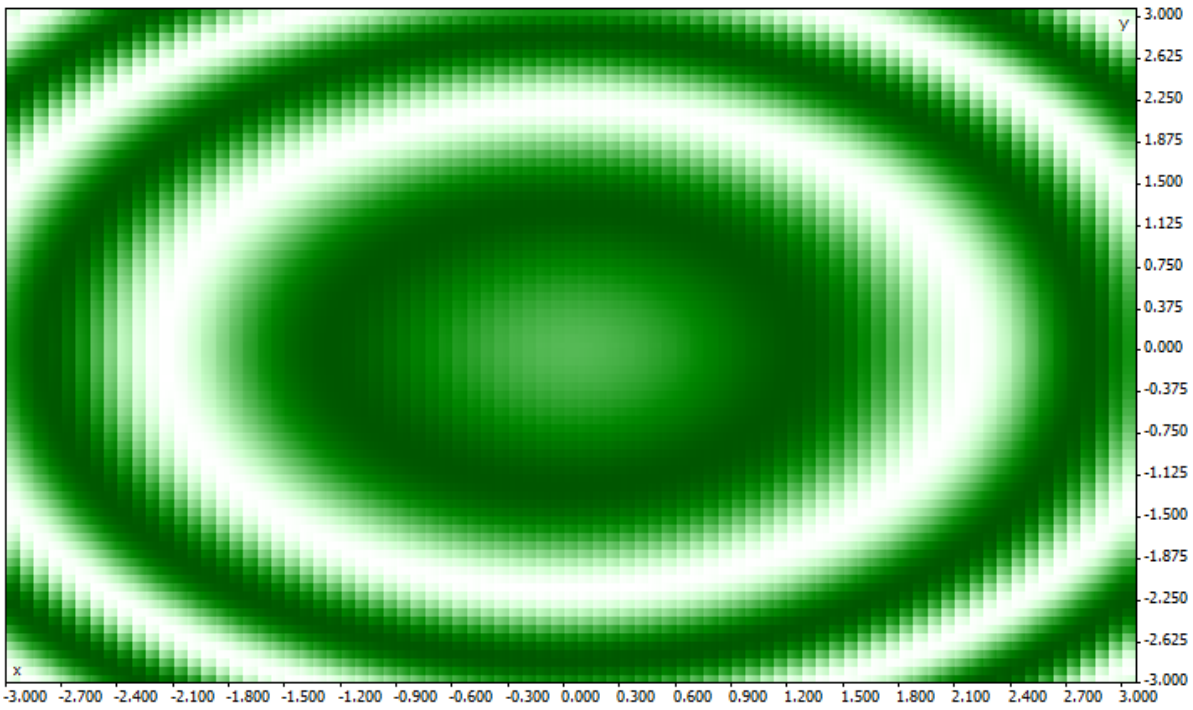
A titre d'exemple, donnons la fonction sink():

$$sink(x,y) = \sin(x^2 + y^2)$$

Plaçons le code du conseiller pour la recherche de l'extremum de cette fonction à OnTester():

```
//+------------------------------------------------------------------+
//|                                                        Sink.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- input parameters
input double   x=-3.0; // start=-3, step=0.05, stop=3
input double   y=-3.0; // start=-3, step=0.05, stop=3
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//---
   double sink=MathSin(x*x+y*y);
//---
   return(sink);
  }
//+------------------------------------------------------------------+
```

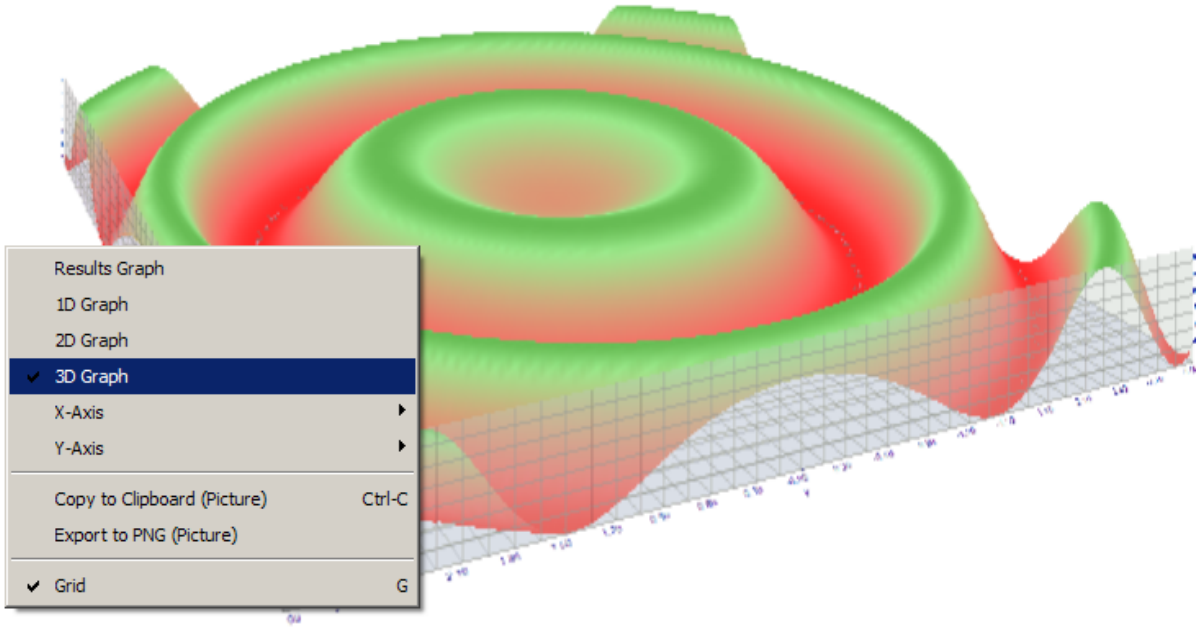
faisons l'optimisation et représenteronsles résultats de l'optimisationen forme du graphique 2D.



Que plus la valeur est mieux pour la paire donnée des paramètres (x, y), plus la couleur est saturée. Comme on s'attendait de l'aspect de la formule de la fonction sink (), ses valeurs forment les cercles concentriques avec le centre dans le point (0,0). Il n'y a pas l'extremum absolu pour la fonction sink ().

© 2000-2015, MetaQuotes Software Corp.

C'est évident à l'affichage des résultats de l'optimisation en mode 3D:



La synchronisation des barres au test en mode "Seulement les prix de l'ouverture"

Le testeur dans le terminal de client MetaTrader 5 permet de vérifier les conseillers soi-disant "multi-devises". Le conseiller multi-devises est un conseiller, qui vend à deux ou plus symboles.

Le test des stratégies vendant sur quelques instruments, impose quelques exigences supplémentaires techniques au testeur:

- de la génération des ticks pour ces instruments;
- le calcul des valeurs des indicateurs pour ces instruments;
- le calcul des exigences de marge selon ces instruments;
- la synchronisation des successions de tick générées selon tous les instruments commercials.

Le testeur génère et perd la succession de tick pour chaque instrument conformément au mode choisi du commerce. En celaune nouvelle barresur chaque instrument s'ouvre indépendamment du fait comment la barre s'est ouverte sur un autre outil. Cela signifie qu'au test de l'expert multi-devises la situation (et le plus souvent c'est comme ça) est possible, quand sur un instrument une nouvelle barre s'est déjà ouverte, et sur l'autre pas encore. Ainsi, au test tout se passe comme dans la vie.

Une telle modélisation authentique du développement de l'histoire dans le testeur ne provoque pas les questions jusqu'à ce qu'on utilise les modes du test "tous les ticks" et "1 minute OHLC". A ces modes dans la limite d'une bougie on génère la quantité suffisante des ticks pour attendre le moment de la synchronisation des barres de symboles différents. Mais comment tester les stratégies multi-devises en mode "Seulement les prix de l'ouverture", si la synchronisation obligatoire des barres sur les instruments commerciales est necéssaire? En effet, en ce mode l'expert est appelé seulement sur un tick, qui correspond au temps de l'ouverture de la barre.

Expliquerons sur l'exemple: si nous testons l'expert sur le symbole EURUSD, et sur EURUSD s'est ouverte une nouvelle bougie horaire, nous apprenons facilement ce fait - au test en mode "Seulement

© 2000-2015, MetaQuotes Software Corp.

les prix de l'ouverture" l'événementNewTickcorrespond au moment de l'ouverture de la barre sur la période testée. Mais il n'y a d'aucune garantie qu'une nouvelle bougie s'est ouverte selon le symbole GBPUSD, qui est utilisé dans l'expert.

Dans les conditions ordinaires il suffit de terminer le travail de la fonction OnTick() et vérifier l'apparition d'une nouvelle barre sur GBPUSD sur le tick suivant. Mais au test en mode "Seulement les prix de l'ouverture" il n'y aura pas un autre tick, et il peut sembler que ce mode ne convient pas au test des experts multi-devises. Mais ce n'est pas vrai - n'oubliez pas que le testeur à MetaTrader 5 se comporte comme dans la vie. On peut attendre le moment, quand sur un autre symbole s'ouvrira une nouvelle barre à l'aide de la fonction Sleep ()!

Le code du conseiller Synchronize_Bars_Use_Sleep.mq5, qui représente l'exemple de la synchronisation des barres au test en mode "Seulement les prix de l'ouverture":

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                  Synchronize_Bars_Use_Sleep.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- input parameters
input string   other_symbol="USDJPY";
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vérifierons le symbole courant
   if(_Symbol==other_symbol)
     {
      PrintFormat("Il est nécessaire d'indiquer un autre symbole ou lancer le test à u
      //--- cessons forcément le test de l'expert
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- la variable statique pour stocker le temps de l'ouverture de la dernière barre
   static datetime last_bar_time=0;
//--- le signe de ce que le temps de l'ouverture de la dernière barre sur les symboles
   static bool synchonized=false;
//--- si la variable statique pas encore initialisée
   if(last_bar_time==0)
     {
      //--- c'est le premier appel, inscrirons le temps de l'ouverture et sortirons
      last_bar_time=(datetime)SeriesInfoInteger(_Symbol,Period(),SERIES_LASTBAR_DATE);
      PrintFormat("On a initialisé la variable last_bar_time par la valeur %s",TimeToS
     }
//--- recevrons le temps de l'ouverture de la dernière barre selon le symbole
   datetime curr_time=(datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_LASTBAR_DAT
//--- si le temps de l'ouverture de la barre actuelle ne correspond pas à celle qui es
   if(curr_time!=last_bar_time)
     {
      //--- retiendrons le temps de l'ouverture de la nouvelle barre dans la variable
      last_bar_time=curr_time;
      //--- la synchronisation est rompue, enlevons le drapeau dans false
      synchonized=false;
      //--- déduisons le message sur cet événement
      PrintFormat("Une nouvelle barre s'est ouverte sur le symbole %s dans %s",_Symbol
     }
//--- garderons ici le temps de l'ouverture de la barre sur le symbole étranger
   datetime other_time;
//--- la boucle jusqu'à ce que le temps de l'ouverture de la dernière barre sur un aut
   while(!(curr_time==(other_time=(datetime)SeriesInfoInteger(other_symbol,Period(),SE
     {
      PrintFormat("Attendrons 5 secondes..");
      //--- attendrons 5 secondes et demanderons encore SeriesInfoInteger(other_symbol
      Sleep(5000);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- le temps de l'ouverture de la barre est identique maintenant pour les deux symbo
   synchonized=true;
   PrintFormat("Le temps de l'ouverture de la dernière barre sur le symbole %s: %s",_S
   PrintFormat("Le temps de l'ouverture de la dernière barre sur le symbole %s: %s",ot
//--- TimeCurrent() ne convient pas, utilisons TimeTradeServer() pour
   Print("Les barres étaient synchronisées dans ",TimeToString(TimeTradeServer(),TIME_
  }
//+------------------------------------------------------------------+
```

Prêtez attention à la dernière ligne dans le conseiller, qui déduit le temps courant, quand le fait de la synchronisation a été établi:

```
   Print("Les barres étaient synchronisées dans ",TimeToString(TimeTradeServer(),TIME_
```

Pour déduire le temps courant nous avons utilisé la fonction TimeTradeServer(), et nonTimeCurrent(). Le fait est que la fonction TimeCurrent () rend le temps du dernier tick, qui n'a pas aucunement changé après l'utilisation de Sleep(). Lancez le conseiller en mode "Seulement les prix de l'ouverture" et verrez les messages sur la synchronisation des barres.

Core 1	2010.12.01 20:00:05	The bars are synchronized at 2010.12.01 20:00:05
Core 1	2010.12.01 20:00:05	Open bar time of the chart symbol USDJPY: 2010.12.01 20:00
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 20:00:00	Waiting 5 seconds..
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 16:00:05	The bars are synchronized at 2010.12.01 16:00:05

Utilisez la fonction TimeTradeServer () au lieu de TimeCurrent (), s'il faut recevoir le temps courant de serveur, et non le temps de l'entrée du dernier tick.

Il y a un autre moyen de la synchronisation des barres - à l'aide de la minuterie. L'exemple d'un tel expert Synchronize_Bars_Use_OnTimer.mq5 est attaché à l'article.

La fonction IndicatorRelease() dans le testeur

Après la fin du test séparé le graphique de l'instrument s'ouvre automatiquement, où les marchés passés et les indicateurs qui étaient utilisés dans l'expert sont affichés. Cela aide à vérifier visuellement les moments de l'entrée et la sortie, ainsi que les comparer aux valeurs des indicateurs.

> **Important:**les indicateurs affichés sur le graphique automatiquement ouvert après l'achèvement du test, sont calculés de nouveau après la fin du test. Même si ces indicateurs étaient utilisés dans l'expert testé.

Mais dans certains cas au programmeur peut être nécessaire de cacher l'information quels indicateurs participent à l'algorithme commercial. Par exemple, le code de l'expert est donné au louage ou est vendu en forme du fichier exécuté sans code initial. Pour cela sera utile la fonction IndicatorRelease().

Si dans le terminal est spécifié le modèle avec le nom tester.tpl dans le répertoire /profiles/templates du terminal de client, notamment il sera appliqué au graphique ouvert. En son absence, le modèle est appliquée par défaut (default.tpl).

La fonctionIndicatorRelease()est initialement destinée à la désallocation de la partie calculée de

© 2000-2015, MetaQuotes Software Corp.

l'indicateur, si c'est nécessaire. Cela permet d'économiser la mémoire, ainsi que les ressources du processeur, parce que chaque tick appelle le calcul de l'indicateur. Sa deuxième destination - c'est interdire l'affichage de l'indicateur sur le graphique du test après la fin du passage unique.

Pour interdire l'affichage de l'indicateur sur le graphique à la fin du test, appelezIndicatorRelease() avec le handle de l'indicateur dans le gestionnaireOnDeinit(). La fonction OnDeinit() est toujours appelée après l'achèvement et avant l'affichage du graphique du test.

```
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   bool hidden=IndicatorRelease(handle_ind);
   if(hidden) Print("IndicatorRelease() est exécuté avec succès ");
   else Print("IndicatorRelease() a rendu false. Le code de l'erreur ",GetLastError())
  }
```

Pour interdire l'affichage de l'indicateur sur le graphique à la fin du test unique, utilisez la fonction IndicatorRelease() dans le gestionnaire OnDeinit().

Le traitement des événements dans le testeur

La présence du gestionnaire OnTick () dans l'expert n'est pas obligatoire pour que l'on pouvait soumettre à la vérification des données historiques dans le testeur du terminal MetaTrader 5. Il suffit que dans le conseiller il y avait quand même une fonction-gestionnaire des énumérées:

- OnTick() - le gestionnaire de l'événement de l'arrivée d'un nouveau tick;
- OnTrade() - le gestionnaire de l'événement commercial;
- OnTimer() - le gestionnaire de l'événement de l'arrivée du signal de la minuterie;
- OnChartEvent() - le gestionnaire des événements d'utilisateur.

On peut traiter les événements d'utilisateur au test dans l'expert à l'aide de la fonctionOnChartEvent(), mais dans les indicateurs cette fonction n'est pas appelée dans le testeur. Même si l'indicateur a le gestionnaireOnChartEvent()et cet indicateur est utilisé dans l'expert de test, l'indicateur lui-même ne recevra pas les événements d'utilisateur.

L'indicateur peut générer les événements d'utilisateur au test à l'aide de la fonction EventChartCustom(), et le conseiller peut traiter cet événement dans OnChartEvent().

Sauf les événements indiqués ci-dessus dans le testeur des stratégies on génère les événements spéciaux liés au procès du test et de l'optimisation:

- Tester - cet événement est généré au terme du test de l'expert aux données historiques. Le traitement de l'événement Tester est produit par la fonction OnTester(). Cette fonction peut être utilisée seulement dans les experts au test et elle est destinée en premier lieu au calcul d'une certaine valeur utilisée à titre du critère Custom max à l'optimisation génétique des paramètres d'entrée.
- TesterInit - cet événement est généré au lancement de l'optimisation dans le testeur des stratégies avant le premier passage. Le traitement de l'événement TesterInit est produit par la fonction OnTesterInit(). L'expert qui a ce gestionnaire, au lancement de l'optimisation est chargé automatiquement sur le graphique séparé du terminal avec le symbole indiqué dans le testeur et la

© 2000-2015, MetaQuotes Software Corp.

période, et reçoit l'événement TesterInit. La fonction est destinée à l'initialisation de l'expert avant l'optimisation pour le traitement suivant des résultats de l'optimisation.

- TesterPass - cet événement est généré à l'entréed'une nouvelle trame des données. Le traitement de l'événement TesterPass est produit par la fonction OnTesterPass(). L'expert qui a ce gestionnaire est chargé automatiquement sur le graphique séparé du terminal avec le symbole/période indiqué pour le test et reçoit les événements TesterPass pendant l'optimisation à l'entrée de la trame. La fonction est destinée au traitement dynamique des résultats de l'optimisation toute de suite, sans attendre son achèvement. L'ajout des trames est faite par la fonction FrameAdd(), qui peut être appelée après un seul passage dans le gestionnaire OnTester().
- TesterDeinit - cet événement est généré à l'issue de l'optimisation de l'expert dans le testeur des stratégies. Le traitement de l'événement TesterDeinit est produit par la fonction OnTesterDeinit(). L'expert qui a ce gestionnaire est chargé automatiquement sur le graphique au lancement de l'optimisation et reçoit l'événement TesterDeinit après son achèvement. La fonction est destinée au traitement final de tousles résultats de l'optimisation.

Les agents du test

Le testeur dans le terminal de client MetaTrader 5 se réalise à l'aide des agents du test. Les agents locaux sont créés et sont connectés automatiquement. Le nombre d'agents locaux correspond par défaut au nombre de noyaux de l'ordinateur.

Chaque agent du test a sa copie des variables globales, qui n'est aucunement liée au terminal de client. Le terminal lui-même est le distributeur, qui distribue les tâches aux agents locaux et distants. Après l'exécution de la tâche suivant selon le test du conseiller avec les paramètres spécifiées l'agent rend les résultats au terminal. Au test séparé on utilise seulement un agent.

L'agent sauvegarde l'histoire reçue du terminal dans les dossiers séparés selon le nom de l'instrument, c'est-à-dire l'histoire pour EURUSD se trouve dans le dossier avec le nom EURUSD. En outre l'histoire des instruments se divise selon les sources. La structure pour le stockage de l'histoire a un tel aspect:

```
le répertoire_du testeur\Agent-IPaddress-Port\bases\le nom_de la source\history\le nom
```

Par exemple, l'histoire selon EURUSD du serveur MetaQuotes-Demo peut se stocker dans le dossier le répertoire_du testeur\Agent-127.0.0.1-3000\bases\MetaQuotes-Demo\EURUSD.

L'agent local après la fin du test se trouve en mode de l'attente de la tâche suivante pendant 5 minutes pour ne pas perdre le temps au lancement aux appels suivants. Seulement à l'expiration de l'attente l'agent local cesse son travail et se décharge de la mémoire de l'ordinateur.

A l'achèvement anticipé du test par l'utilisateur (le bouton "Annulation"), ainsi qu'à la clôture du terminal de client tous les agents locaux cessent le travail et se déchargent de la mémoire.

L'échange des données entre le terminal et l'agent

Au lancement du test le terminal prépare quelques blocs des paramètres pour l'expédition à l'agent:

- Les paramètres d'entrée du test (les modes de la modélisation, l'intervalle du test, l'instrument, le critère de l'optimisation etc.)
- La liste des sélectionnés dans "Aperçu du marché" des instruments
- La spécification de l'instrument testé (la taille du contrat, les alinéas admissibles du marché pour l'installation de StopLoss et Takeprofit, etc)

© 2000-2015, MetaQuotes Software Corp.

- L'expert testé et les valeurs de ses paramètres d'entrée
- L'information sur les fichiers supplémentaires (les bibliothèques, les indicateurs, les fichiers des données - #property tester_...)

tester_indicator	string	Le nom de l'indicateur d'utilisateur dans le format "le nom_de l'indicateur.ex5". Les indicateurs nécessaires au test sont définis automatiquement de l'appel des fonctions iCustom(), si le paramètre correspondant est spécifié par la chaîne constante. Cette propriété est nécessaire pour les autres cas (l'utilisation de la fonctionIndicatorCreate()ou l'utilisation de la chaîne non constante dans le paramètre qui spécifie le nom de l'indicateur)
tester_file	string	Le nom du fichier pour le testeur avec l'indication de l'extension, conclue aux guillemets doubles (comme la chaîne constante). Le fichier indiqué sera transmis au travail au testeur. Les fichiers d'entrée pour le test, s'ils sont nécessaires, doivent être toujours spécifiés
tester_library	string	Le nom de la bibliothèque avec l'extension, conclue aux guillemets doubles. La bibliothèque peut être avec l'extension dll et ex5. Les bibliothèques nécessaires au test sont définies automatiquement. Cependant, si quelque bibliothèque est utilisée par l'indicateur d'utilisateur, il est nécessaire d'utiliser la propriété donnée

Pour chaque bloc des paramètres il y a l'empreinte digitale en forme de hachage MD5, qui est envoyé à l'agent. Le hachage MD5 est unique pour chaque ensemble, son volume est plus petit que le volume de l'information, à la base de laquelle il est calculé.

L'agent reçoit les hachages des blocs et compare avec ceux qu'il stocke chez lui-même. Si l'empreinte de ce bloc des paramètres manque chez l'agent, ou le hachage envoyé se distingue de l'actuel, l'agent demande le bloc des paramètres. Ceci permet de réduire le trafic entre le terminal et l'agent.

© 2000-2015, MetaQuotes Software Corp.

Après le test l'agent rend au terminal tous les résultats du passage montrés dans les onglets "Résultats du test" et "Résultats de l'optimisation": le bénéfice réalisé, la quantité de marchés, le ratio de Sharpe, le résultat de la fonction OnTester() etc.

A l'optimisation le terminal distribue aux agents les tâches pour faire le test par les petits paquets, dans chaque paquet il y a quelques tâches (chaque tâche signifie le test séparé avec l'ensemble des paramètres d'entrée). Cela diminue le temps de l'échange entre le terminal et l'agent.

Les agents n'enregistrent jamais sur le disque dur les fichiers EX5 reçus du terminal (l'expert, les indicateurs, les bibliothèques etc.) des considérations de la sécurité pour qu'on ne pouvait pas se servir par les données envoyés sur l'ordinateur avec l'agent installé. Tous les autres fichiers, y compris DLL, s'enregistrent au "sandbox" Dans les agents distants on ne peut pas tester les experts avec l'utilisation de DLL.

Les résultats du test se forment par le terminal à la cache spéciale des résultats (la cache résultant) pour l'accès ultérieur rapide à eux en cas de nécessité. Pour chaque ensemble des paramètres le terminal cherche dans la cache résultant les résultats déjà prêts des mises en marche précédentes pour excepter les redémarrages. Si le résultat avec un tel ensemble des paramètres n'est pas trouvé, on donne le devoir pour faire le test à l'agent.

Tout le trafic entre le terminal et les agents est chiffré.

> Les ticks ne sont pas envoyés par le réseau, ils sont générés sur les agents de test.

L'utilisation du dossier commun de tous les terminaux de client

Tout les agents de test sont isolés l'un de l'autre et du terminal de client: chaque agent a un dossier personnel, où s'inscrivent les logs de l'agent. En outre toutes les opérations de fichier au test de l'agent se passent dans le dossier **le nom_de l'agent/MQL5/Files**. On peut réaliser cependant la coopération entre les agents locaux et le terminal de client dans le dossier total de tous les terminaux de client, si on indique le drapeau FILE_COMMONà l'ouverture du fichier:

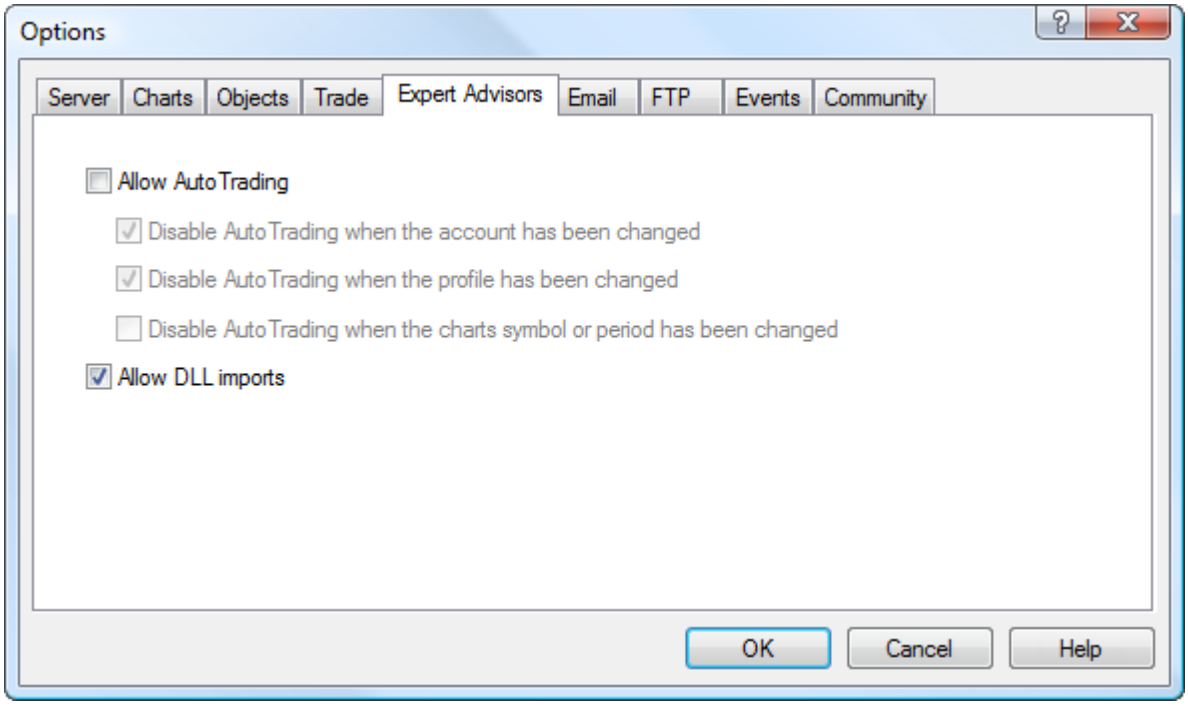
```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le dossier total de tous les terminaux de client
   common_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- déduisons le nom de ce dossier
   PrintFormat("Ouvrirons le fichier dans le dossier total des terminaux de client %s'
//--- ouvrirons le fichier dans le dossier total (le drapeau FILE_COMMON est indiqué)
   handle=FileOpen(filename,FILE_WRITE|FILE_READ|FILE_COMMON);
  ... les actions ultérieures
//---
   return(INIT_SUCCEEDED);
  }
```

L'utilisation de DLL

© 2000-2015, MetaQuotes Software Corp.

Pour l'accélération de l'optimisation on peut utiliser non seulement les agents locaux, mais aussi les agents distants. Et il y a des limites aux agents distants. Premièrement, les agents distants ne déduisent pas les résultats de l'exécution de la fonction Print()dans ses logs, les messages sur l'ouverture et la clôture des positions. Le minimum de l'information est déduit dans le log pour que les experts incorrectement écrits ne remplissent le disque dur de l'ordinateur pas par les messages, sur lequel l'agent distant travaille.

La deuxième restriction - c'est l'interdiction à l'utilisation de DLL au test des experts. Les appels DLL sont absolument interdits sur les agents distants de la considération de la sécurité. Sur les agents locaux les appels dll dans les experts testés sont permis seulement dans le cas s'il y a la permission correspondante "Permettre l'importation de DLL".



Considérablement:à l'utilisation (des scripts, des indicateurs) reçus des conseillers, qui demandent de permettre les appels DLL, vous devez vous rendre compte de tout le risque assumé en cas de la permission de cette option dans les réglages du terminal. Indépendamment du fait comment le conseiller sera utilisé - pour le test ou pour le lancement sur le graphique.

© 2000-2015, MetaQuotes Software Corp.

Les variables prédéterminées

Pour chaque programme mql5 exécuté on soutient une série de variables prédéterminées, qui reflètent l'état du graphique courant de prix au moment du lancement du programme - l'expert, le script ou l'indicateur d'utilisateur.

Le terminal de client établit la valeur aux variables prédéterminées avant le lancement du programme mql5 à l'exécution. Les variables prédéterminées sont constantes et ne peuvent pas être changées du programme mql5. L'exception est faite pour la variable _LastError, qui peut être remis au zéro par la fonction ResetLastError.

Variable	Valeur
_Digits	Le nombre de signes décimaux après la virgule
_Point	La grandeur du point de l'instrument courant en devise de la cotation
_LastError	La signification de la dernière erreur
_Period	La signification du temps trame du graphique courant
_RandomSeed	L'état actuel du générateur des nombres entiers aléatoires
_StopFlag	Le drapeau de l'arrêt du programme
_Symbol	Le nom du symbole du graphique courant
_UninitReason	Le code de la raison de la déinitialisation du programme

Les bibliothèques se servent des variables du programme qui les a appelées.

© 2000-2015, MetaQuotes Software Corp.

int _Digits

Dans la variable _Digits se trouve le nombre de signes décimaux après la virgule, qui définit l'exactitude de la mesure du prix du symbole du graphique courant.

On peut utiliser aussi la fonction Digits().

© 2000-2015, MetaQuotes Software Corp.

double _Point

Dans la variable _Point se trouve la grandeur de point du symbole actuel dans la devise de la cotation.

On peut utiliser aussi la fonction Point().

© 2000-2015, MetaQuotes Software Corp.

int _LastError

Dans la variable _LastError se trouve se trouve la signification de la dernière erreur, passé pendant l'exécution du programme mql5. On peut remettre la valeur au zéro par la fonction ResetLastError().

Pour obtenir le code de la dernière erreur, on peut utiliser aussi la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

int _Period

Dans la variable _Period se trouve la signification du temps trame du graphique courant.

On peut utiliser aussi la fonction Period().

Voir aussi

PeriodSeconds, Les périodes des graphiques, La date et le temps, La visibilité des objets

© 2000-2015, MetaQuotes Software Corp.

_RandomSeed

La variable pour stocker l'état actuel à la génération des nombres entiers aléatoires. _RandomSeedchange la valeur à l'appel de MathRand(). UtilisezMathSrand() pour définir l'état initial nécessaire.

Un nombre aléatoire ***x***, reçu par la fonction MathRand(), est calculé à chaque appel comme ça:

```
x=_RandomSeed*214013+2531011;
_RandomSeed=x;
x=(x>>16)&0x7FFF;
```

Voir aussi

MathRand(), MathSrand(), Les types entiers

© 2000-2015, MetaQuotes Software Corp.

bool _StopFlag

Dans la variable _StopFlag se trouve le drapeau de l'arrêt du programme mql5. Quand le terminal de client tente d'arrêter le programme, la valeur true s'inscrit à cette variable.

Pour vérifier l'état du drapeau _StopFlag on peut aussi utiliser la fonction IsStopped().

© 2000-2015, MetaQuotes Software Corp.

string _Symbol

Dans la variable _Symbol se trouve le nom du symbole du graphique courant.

On peut utiliser aussi la fonction Symbol().

© 2000-2015, MetaQuotes Software Corp.

int _UninitReason

Dans la variable _UninitReason se trouve le code de la raison de la déinitialisation du programme.

D'habitude on reçoit le code de la raison de la déinitialisation à l'aide de la fonction UninitializeReason().

© 2000-2015, MetaQuotes Software Corp.

Les fonctions communes

Les fonctions de la destination commune, qui ne sont pas entrées dans un des groupes spécialisés.

Fonction	Action
Alert	Affiche le message dans la fenêtre séparée
CheckPointer	Rend le type de l'indication de l'objet
Comment	Affiche le message à un angle gauche supérieur du graphique de prix
CryptEncode	Преобразует данные массива-источника в массив-приемник указанным методом
CryptDecode	Производит обратное преобразование данных массива
DebugBreak	Le point d'arrêt de programme de débogage
ExpertRemove	Arrête le travail de l'expert et le décharge du graphique
GetPointer	Rend l'indicateur de l'objet
GetTickCount	Rend la quantité des millisecondes qui ont passé dès le moment du départ du système
GetMicrosecondCount	Возвращает количество микросекунд, прошедших с момента начала работы MQL5-программы
MessageBox	Crée, affiche la fenêtre des messages et la dirige
PeriodSeconds	Rend la quantité de secondes dans la période
PlaySound	Reproduit le fichier sonore
Print	Affiche un message dans le journal
PrintFormat	Formate et imprime l'ensemble des symboles et des valeurs dans un fichier-journal conformément au format donné
ResetLastError	Établit la valeur de la variable prédéterminée _LastError au zéro
ResourceCreate	Crée la ressource de l'image à la base de l'ensemble des données
ResourceFree	Supprime la ressource dynamiquement créée (libère la mémoire occupée par la ressource)
ResourceReadImage	Lit les données de la ressource graphique, créé par la fonction ResourceCreate() ou sauvegardée dans le fichier EX5 à la compilation

© 2000-2015, MetaQuotes Software Corp.

ResourceSave	Sauvegarde la ressource au fichier spécifié
SendFTP	Envoie le fichier à l'adresse indiquée dans la fenêtre des réglages du signet "la publication"
SendMail	Envoie le message électronique à l'adresse indiquée dans la fenêtre des réglages du signet "la Poste"
SendNotification	Envoie les notifications Push aux terminaux mobiles, dont les ID MetaQuotes sont indiqués sur l'onglet "Notifications"
Sleep	Retient l'exécution de l'expert courant ou le script sur l'intervalle défini
TerminalClose	Envoie au terminal l'ordre sur l'achèvement du travail
TesterStatistics	Rend la valeur du paramètre indiqué statistique, calculée d'après les résultats du test
WebRequest	Отправляет HTTP-запрос на указанный сервер
ZeroMemory	Remet au zéro la variable transmise selon la référence. La variable peut être de n'importe quel type, seulement les classes et les structures qui ont les constructeurs font l'exception

© 2000-2015, MetaQuotes Software Corp.

Alert

Affiche la fenêtre de dialogue, contenant les données d'utilisateur.

```
void  Alert(
   argument,      // première valeur
   ...            // valeurs ultérieures
   );
```

Paramètres

argument

[in] N'importe quelles valeurs divisées par les virgules. Pour la division de l'information déduite en quelques lignes on peut utiliser le symbole de la transmission de la ligne "\n" ou "\r\n". La quantité de paramètres ne peut pas excéder 64.

La valeur rendue

Il n'y a pas de valeur rendue

Note

On ne peut pas transmettre les tableaux à la fonction Alert(). Les tableaux doivent être déduits par les éléments. Les données de type double sont déduites avec 8 chiffres décimaux après le point, les données de type float sont déduites avec 5 chiffres décimaux après le point. Pour la sortie des nombres réels avec une autre exactitude ou dans le format scientifique il est nécessaire d'utiliser la fonction DoubleToString().

Les données du type bool sont déduites en forme des chaînes "true" ou "false". Les dates sont déduites comme YYYY.MM.DD HH:MI:SS. Pour la sortie de la date dans un autre format il est nécessaire d'utiliser la fonction TimeToString(). Les données du type color sont déduites en forme de la chaîne R,G,B, ou en forme du nom de la couleur, si cette couleur assiste dans l'ensemble des couleurs.

При работе в тестере стратегий функция Alert() не выполняется.

© 2000-2015, MetaQuotes Software Corp.

CheckPointer

Rend le type de l'indicateur de l'objet.

```
ENUM_POINTER_TYPE  CheckPointer(
   object* anyobject      // indicateur de l'objet
   );
```

Paramètres

anyobject

[in] L'indicateur de l'objet.

La valeur rendue

Rend la valeur de l'énumération ENUM_POINTER_TYPE.

Note

La tentative de l'appel à l'indicateur incorrect amène à l'achèvement critique du programme. C'est pourquoi il y a une nécessité de l'utilisation de la fonction CheckPointer avant l'utilisation de l'indicateur. L'indicateur peut être incorrect dans les cas suivants:

- l'indicateur est égal à NULL;
- si l'objet était supprimé à l'aide de l'opérateur delete.

On peut utiliser cette fonction le contrôle de l'indicateur sur la validité. La valeur, distinguée du zéro, garantit qu'on peut avoir l'accès selon l'indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//| Supprimer la liste en suppriment ses éléments                    |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- indicateur de service pour le travail dans la boucle
   CItem* item;
//--- passons par la boucle et essayez de supprimer les indicateurs dynamiques
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("Dynamic object ",item.Identifier()," to be deleted");
         delete (item);
        }
      else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
     }
//---
  }
```

Voir aussi

Les indicateurs des objets, Le contrôle de l'indicateur de l'objet, L'opérateur de l'effacement de l'objet delete

© 2000-2015, MetaQuotes Software Corp.

Comment

Déduit le commentaire défini par l'utilisateur, à un angle gauche supérieur du graphique.

```
void  Comment(
   argument,      // première valeur
   ...            // valeurs ultérieures
   );
```

Paramètres

...

[in] N'importe quelles valeurs divisées par les virgules. Pour la division de l'information déduite en quelques lignes on peut utiliser le symbole de la transmission de la ligne "\n" ou "\r\n". La quantité de paramètres ne peut pas excéder 64. La longueur totale le message déduit (y compris les symboles de service invissibles) ne peut pas excéder 2045 symboles (les symboles superflus seront coupés à la sortie.

La valeur rendue

Il n'y a pas de valeur rendue

Note

On ne peut pas transmettre les tableaux à la fonction Comment(). Les tableaux doivent être imprimés par les éléments.

Les données du type double sont déduites à près de 16 chiffres décimaux après le point, de plus les données peuvent être déduites au format traditionnel ou dans le format scientifique – suivant quelle inscription sera plus compacte. Les données du type float sont déduites avec 5 chiffres décimaux après le point. Pour la sortie des nombres réels avec une autre exactitude ou dans le format spécifié il est nécessaire d'utiliser la fonction DoubleToString().

Les données du type bool sont déduites en forme des chaînes "true" et "false". Les dates sont déduites comme YYYY.MM.DD HH:MI:SS. Pour la sortie de la date dans un autre format il est nécessaire d'utiliser la fonction TimeToString(). Les données du type color sont déduites en forme de la chaîne R,G,B, ou en forme du nom de la couleur, si cette couleur assiste dans l'ensemble des couleurs.

Exemple:

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
//--- déduisons les valeurs à trois lignes
   Comment(StringFormat("déduisons les prix\nAsk = %G\nBid = %G\nSpread = %d",Ask,Bid,
  }
```

© 2000-2015, MetaQuotes Software Corp.

Voir aussi

ChartSetString, ChartGetString

© 2000-2015, MetaQuotes Software Corp.

CryptEncode

Преобразует данные массива-источника в массив-приемник указанным методом.

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,        // метод преобразования
   const uchar&        data[],        // массив-источник
   const uchar&        key[],         // ключ шифрования
   uchar&              result[]       // массив-приемник
   );
```

Параметры

method

[in] Метод преобразования. Может быть одним из значений перечисления ENUM_CRYPT_METHOD.

data[]

[in] Массив-источник.

key[]

[in] Ключ шифрования.

result[]

[out] Массив-приемник.

Возвращаемое значение

Количество байт в массиве-приемнике или 0 в случае ошибки. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";
//--- проверка размера
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);
//--- преобразование в шестнадцатиричную строку
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
   string text="The quick brown fox jumps over the lazy dog";
   string keystr="ABCDEFG";
   uchar src[],dst[],key[];
//--- подготовка ключа шифрования
   StringToCharArray(keystr,key);
//--- подготовка исходного массива src[]
   StringToCharArray(text,src);
//--- вывод исходных данных
   PrintFormat("Initial data: size=%d, string='%s'",ArraySize(src),CharArrayToString(s
//--- шифрование массива src[] методом DES с 56-битным ключом key[]
   int res=CryptEncode(CRYPT_DES,src,key,dst);
//--- проверка результата шифрования
   if(res>0)
     {
      //--- вывод шифрованных данных
      PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
      //--- расшифровка данных массива dst[] методом DES с 56-битным ключом key[]
      res=CryptDecode(CRYPT_DES,dst,key,src);
      //--- проверка результата
      if(res>0)
        {
         //--- вывод дешифрованных данных
         PrintFormat("Decoded data: size=%d, string='%s'",ArraySize(src),CharArrayToSt
        }
      else
         Print("Ошибка в CryptDecode. Код ошибки=",GetLastError());
     }
   else
      Print("Ошибка в CryptEncode. Код ошибки=",GetLastError());
  }
```

Смотри также

Операции с массивами, CryptDecode()

© 2000-2015, MetaQuotes Software Corp.

CryptDecode

Производит обратное преобразование данных массива, полученного при помощи функции CryptEncode().

```
int  CryptDecode(
   ENUM_CRYPT_METHOD   method,        // метод преобразования
   const uchar&        data[],        // массив-источник
   const uchar&        key[],         // ключ шифрования
   uchar&              result[]       // массив-приемник
   );
```

Параметры

method

[in] Метод преобразования. Может быть одним из значений перечисления ENUM_CRYPT_METHOD.

data[]

[in] Массив-источник.

key[]

[in] Ключ шифрования.

result[]

[out] Массив-приемник.

Возвращаемое значение

Количество байт в массиве-приемнике или 0 в случае ошибки. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

Смотри также

Операции с массивами, CryptEncode()

© 2000-2015, MetaQuotes Software Corp.

DebugBreak

Le point d'arrêt de programme de débogage.

```
void  DebugBreak();
```

La valeur rendue

Il n'y a pas de valeur rendue.

Note

L'interruption de l'exécution du programme mql5 se passe seulement dans le cas où le programme est lancé en régime du réglage. On peut utiliser la fonction pour l'affichage des valeurs des variables et/ou une exécution ultérieure pas à pas.

© 2000-2015, MetaQuotes Software Corp.

ExpertRemove

Arrête le travail de l'expert et la décharge du graphique.

```
void  ExpertRemove();
```

La valeur rendue

Il n'y a pas de valeur rendue.

Note

L'arrêt de l'expert ne se passe pas immédiatement à l'appel de la fonction ExpertRemove(), on produit seulement la section du drapeau pour l'arrêt du travail de l'expert. C'est-à-dire, l'expert ne traitera pas déjà un événement suivant, on produira l'appel OnDeinit() et l'effacement avec l'éloignement du graphique.

Exemple:

```
//+------------------------------------------------------------------+
//|                                              Test_ExpertRemove.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
input int ticks_to_close=20;// nombre de ticks avant la décharge de l'expert
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(TimeCurrent(),": " ,__FUNCTION__," reason code = ",reason);
//--- "clear" comment
   Comment("");
//---
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   static int tick_counter=0;
//---
   tick_counter++;
   Comment("\nAvant la décharge de l'expert ",__FILE__," reste ",
           (ticks_to_close-tick_counter)," des ticks");
//--- jusqu'au
   if(tick_counter>=ticks_to_close)
     {
      ExpertRemove();
      Print(TimeCurrent(),": ",__FUNCTION__," expert sera déchargé ");
     }
   Print("tick_counter = ",tick_counter);
//---
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

Voir aussi

L'exécution des programmes, Les événements du terminal de client

© 2000-2015, MetaQuotes Software Corp.

GetPointer

Rend l'indicateur de l'objet.

```
void*  GetPointer(
   any_class anyobject      // objet de n'importe quelle classe
   );
```

Paramètres

anyobject

[in] L'objet de n'importe quelle classe.

La valeur rendue

Rend l'indicateur de l'objet.

Note

Seulement les objets des classes ont les indicateurs. Les exemplaires des structures et les variables des types simples des indicateurs n'ont pas des indicateurs. L'objet de la classe, qui n'est pas créé à l'aide de l'opérateur new(), mais par exemple, automatiquement créé dans le tableau des objets, a l'indicateur tout de même. Seulement cet indicateur aura le type automatique POINTER_AUTOMATIC,et on ne peut pas lui appliquer l'opérateur delete(). Pour le reste l'indicateur du type ne se distingue pas des indicateurs dynamiques du type POINTER_AUTOMATIC.

Puisque les variables comme les structures et les types simples n'ont pas les indicateurs, il est interdit d'appliquer la fonction GetPointer() à eux. Il est aussi interdit de transmettre l'indicateur comme l'argument de la fonction. Dans tous les cas énumérés le compilateur notifiera d'une erreur.

La tentative de l'appel à l'indicateur incorrect amène à l'achèvement critique du programme. C'est pourquoi il y a une nécessité de l'utilisation de la fonction CheckPointer() avant l'utilisation de l'indicateur. L'indicateur peut être incorrect dans les cas suivants:

- l'indicateur est égal à NULL;
- si l'objet était supprimé à l'aide de l'opérateur delete.

On peut utiliser cette fonction le contrôle de l'indicateur sur la validité. La valeur, distinguée du zéro, garantit qu'on peut avoir l'accès selon l'indicateur.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                              Check_GetPointer.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

//+------------------------------------------------------------------+
//| Classe en exécutant l'élément de liste                           |
//+------------------------------------------------------------------+
class CItem
  {
   int               m_id;
   string            m_comment;
   CItem*            m_next;
public:
                     CItem() { m_id=0; m_comment=NULL; m_next=NULL; }
                    ~CItem() { Print("Destructor of ",m_id,
                                     (CheckPointer(GetPointer(this))==POINTER_DYNAMIC)
                                     "dynamic":"non-dynamic"); }
   void              Initialize(int id,string comm) { m_id=id; m_comment=comm; }
   void              PrintMe() { Print(__FUNCTION__,":",m_id,m_comment); }
   int               Identifier() { return(m_id); }
   CItem*            Next() {return(m_next); }
   void              Next(CItem *item) { m_next=item; }
  };
//+------------------------------------------------------------------+
//| Classe la plus simple de la liste                                |
//+------------------------------------------------------------------+
class CMyList
  {
   CItem*            m_items;
public:
                     CMyList() { m_items=NULL; }
                    ~CMyList() { Destroy(); }
    bool             InsertToBegin(CItem* item);
    void             Destroy();
  };
//+------------------------------------------------------------------+
//| Insérer l'élément de liste au commencement                       |
//+------------------------------------------------------------------+
bool CMyList::InsertToBegin(CItem* item)
  {
   if(CheckPointer(item)==POINTER_INVALID) return(false);
//---
   item.Next(m_items);
   m_items=item;
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Supprimer la liste en supprimant des éléments                    |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- indicateur de service pour travailler dans une boucle
   CItem* item;
//--- passons par la boucle et essayez de supprimer les indicateurs dynamiques
   while(CheckPointer(m_items)!=POINTER_INVALID)
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       item=m_items;
       m_items=m_items.Next();
       if(CheckPointer(item)==POINTER_DYNAMIC)
         {
          Print("Dynamyc object ",item.Identifier()," to be deleted");
          delete (item);
         }
       else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
      }
//---
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   CMyList list;
   CItem   items[10];
   CItem*  item;
//--- créons et ajouteront à la liste l'indicateur dynamique de l'objet
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(100,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- ajoutons les indicateurs automatiques à la liste
   for(int i=0; i<10; i++)
     {
      items[i].Initialize(i,"automatic");
      items[i].PrintMe();
      item=GetPointer(items[i]);
      if(CheckPointer(item)!=POINTER_INVALID)
         list.InsertToBegin(item);
     }
//--- ajoutons encore un indicateur dynamique de l'objet au début de la liste
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(200,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//---  supprimons les éléments de la liste
   list.Destroy();
//---  tous les éléments de la liste seront supprimés,
//--- regarde le signet Experts dans le terminal
  }
```

Voir aussi

Les indicateurs des objets, Le contrôle de l'indicateur de l'objet , L'opérateur de l'effacement de l'objet delete

© 2000-2015, MetaQuotes Software Corp.

GetTickCount

La fonction GetTickCount() rend la quantité des millisecondes qui ont passé depuis le début de système.

```
uint  GetTickCount();
```

La valeur rendue

La valeur du type uint.

Note

Le compteur est limité par la résolution systémique de minuteur. Puisque le temps se stocke comme l'entier sans signes, il déborde tous les 49.7 jours au travail continu de l'ordinateur.

Exemple:

```
#define MAX_SIZE 40
//+------------------------------------------------------------------+
/ / | le script pour mesurer le temps de calcul de 40 nombres de Fibonacci |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- rappelons la valeur initiale
   uint start=GetTickCount();
//--- la variable pour la réception du numéro suivant dans la suite de Fibonacci
   long fib=0;
//--- la boucle dans laquelle on calcule un certain nombre de numéros dans la suite de
   for(int i=0;i<MAX_SIZE;i++) fib=TestFibo(i);
//--- recevrons le temps dépensé en millisecondes
   uint time=GetTickCount()-start;
//--- déduisons au journal "Experts" le message
   PrintFormat("Le calcul %d des premiers nombres Fibonacci a pris %d ms",MAX_SIZE,tim
//--- le travail du script est terminé
   return;
  }
//+------------------------------------------------------------------+
//| La fonction de la réception du nombre de Fibonacci selon son numéro d'ordre    |
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- le premier membre de la suite Fibonacci
   if(n<2) return(1);
//--- tous les membres suivants sont calculés par cette formule
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

Voir aussi

Date et heure

© 2000-2015, MetaQuotes Software Corp.

GetMicrosecondCount

Функция GetMicrosecondCount() возвращает количество микросекунд, прошедших с момента начала работы MQL5-программы.

```
ulong  GetMicrosecondCount();
```

La valeur rendue

La valeur du type ulong.

Exemple:

```
//+------------------------------------------------------------------+
//| Тестируемый код                                                  |
//+------------------------------------------------------------------+
void Test()
  {
   int    res_int=0;
   double res_double=0;
//---
   for(int i=0;i<10000;i++)
     {
      res_int+=i*i;
      res_int++;
      res_double+=i*i;
      res_double++;
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   uint   ui=0,ui_max=0,ui_min=INT_MAX;
   ulong  ul=0,ul_max=0,ul_min=INT_MAX;
//--- количество тестов
   for(int count=0;count<1000;count++)
     {
      uint  ui_res=0;
      ulong ul_res=0;
      //---
      for(int n=0;n<2;n++)
        {
         //--- выбираем способ измерения
         if(n==0) ui=GetTickCount();
         else     ul=GetMicrosecondCount();
         //--- тестируемый код
         Test();
         //--- копим результат измерения (в зависимости от способа)
```

© 2000-2015, MetaQuotes Software Corp.

```
         if(n==0) ui_res+=GetTickCount()-ui;
         else     ul_res+=GetMicrosecondCount()-ul;
        }
      //--- собираем минимальное и максимальное время исполнения кода обоих измерений
      if(ui_min>ui_res) ui_min=ui_res;
      if(ui_max<ui_res) ui_max=ui_res;
      if(ul_min>ul_res) ul_min=ul_res;
      if(ul_max<ul_res) ul_max=ul_res;
     }
//---
   Print("GetTickCount error(msec): ",ui_max-ui_min);
   Print("GetMicrosecondCount error(msec): ",DoubleToString((ul_max-ul_min)/1000.0,2));
  }
```

Voir aussi

Date et heure

© 2000-2015, MetaQuotes Software Corp.

MessageBox

Crée et affiche la fenêtre des messages et le dirige. La fenêtre des messages contient le message et le titre, n'importe quelle combinaison des signes prédéterminés et les boutons de commande.

```
int  MessageBox(
   string  text,              // texte de message
   string  caption=NULL,      // titre de la fenêtre
   int     flags=0            // définit l'ensemble des boutons dans la fenêtre
   );
```

Paramètres

text

[in] Le texte contenant le message pour l'affichage.

caption=NULL

[in] Le texte non obligatoire pour l'affichage dans le titre de la fenêtre du message. Si ce paramètre vide, le nom de l'expert sera affichée dans le titre de la fenêtre.

flags=0

[in] Les drapeaux non obligatoires définissant l'aspect et le comportement de la fenêtre de dialogue. Les drapeaux peuvent être la combinaison du groupe spécial des drapeaux.

La valeur rendue

Si la fonction est accomplie avec succès, la valeur rendue - une des valeurs des codes du retour MessageBox().

Note

On ne peut pas appeler la fonction des indicateurs d'utilisateur, puisque les indicateurs sont accomplis dans le thread d'interface et ne doivent pas le ralentir.

При работе в тестере стратегий функция MessageBox() не выполняется.

© 2000-2015, MetaQuotes Software Corp.

PeriodSeconds

Rend la quantité de secondes dans la période.

```
int  PeriodSeconds(
   ENUM_TIMEFRAMES  period=PERIOD_CURRENT      // période du graphique
   );
```

Paramètres

period=PERIOD_CURRENT

[in] La valeur de la période du graphique de l'énumération ENUM_TIMEFRAMES. Si le paramètre n'est pas spécifié, il rend le nombre de secondes de la période de graphique actuelle, à laquelle le programme est lancé.

La valeur rendue

La quantité de secondes dans la période indiquée.

Voir aussi

_Period, Les périodes des graphiques, La date et le temps, La visibilité des objets

© 2000-2015, MetaQuotes Software Corp.

PlaySound

Reproduit le fichier sonore.

```
bool  PlaySound(
   string  filename      // nom du fichier
   );
```

Paramètres

filename

[in] Le chemin vers le fichier sonore. Если filename=NULL, воспроизведение звука прекращается.

La valeur rendue

true - si le fichier sonore est trouvé, autrement rend false.

Note

Le fichier doit être disposé dans le répertoire le répertoire_du terminal\Sounds ou son sous-catalogue. On reproduit seulement les fichiers sonores dans le format WAV.

Вызов PlaySound() с параметром NULL останавливает воспроизведение звука.

При работе в тестере стратегий функция PlaySound() не выполняется.

Voir aussi

Ressources

© 2000-2015, MetaQuotes Software Corp.

Print

Imprime un certain message au journal des experts. Les paramètres peuvent avoir n'importe quel type.

```
void  Print(
   argument,      // première valeur
   ...            // valeurs ultérieures
   );
```

Paramètres

. . .

[in] N'importe quelles valeurs divisées par les virgules.La quantité de paramètres ne peut pas excéder 64.

Note

On ne peut pas transmettre les tableaux à la fonction Print(). Les tableaux doivent être déduits par les éléments.

Les données de type double sont déduites à près de 16 chiffres décimaux après le point, de plus les données peuvent être déduites au format traditionnel ou dans le format scientifique – suivant quelle inscription sera plus compacte. Les données du type float sont déduites avec 5 chiffres décimaux après le point. Pour la sortie des nombres réels avec une autre exactitude ou dans le format spécifié il est nécessaire d'utiliser la fonction PrintFormat().

Les données du type bool sont déduites en forme des chaînes "true" et "false". Les dates sont déduites comme YYYY.MM.DD HH:MI:SS. Pour la sortie de la date dans un autre format il est nécessaire d'utiliser la fonction TimeToString(). Les données du type color sont déduites en forme de la chaîne R,G,B, ou en forme du nom de la couleur, si cette couleur assiste dans l'ensemble des couleurs.

При работе в тестере стратегий в режиме оптимизации функция Print() не выполняется.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
void OnStart()
  {
//--- déduisons DBL_MAX à l'aide de Print(), c'est égal à PrintFormat(%%.16G,DBL_MAX)
   Print("---- comment apparaît DBL_MAX -----");
   Print("Print(DBL_MAX)=",DBL_MAX);
//---maintenant nous déduisons le nombre DBL_MAX à l'aide de PrintFormat()
   PrintFormat("PrintFormat(%%.16G,DBL_MAX)=%.16G",DBL_MAX);
//--- Déduit au journal "Experts"
// Print(DBL_MAX)=1.797693134862316e+308
// PrintFormat(%.16G,DBL_MAX)=1.797693134862316E+308

//--- regardons comment on déduit le type float
   float c=(float)M_PI; // il faut évidemment rendre égal au type cible
   Print("c=",c, "    Pi=",M_PI, "    (float)M_PI=",(float)M_PI);
// c=3.14159    Pi=3.141592653589793    (float)M_PI=3.14159

//--- montrons ce qui peut se passer aux opérations arithmétiques sur les types réels
   double a=7,b=200;
   Print("---- avant les opérations arithmétiques");
   Print("a=",a,"   b=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- devisons a sur b (7/200)
   a=a/b;
//---comme si nous restaurons la valeur dans la variable b maintenant
   b=7.0/a; // il est prévu que b=7.0/(7.0/200.0)=>7.0/7.0*200.0=200 - mais ce n'est p
//--- déduisons une nouvelle valeur calculée b
   Print("-----après les opérations arithmétiques");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- déduit au journal "Experts"
// Print(b)=200.0
// Print(DoubleToString(b,16))=199.9999999999999716 ( voyons qu'en fait b n'est pas éq

//--- créons une très petite valeur epsilon=1E-013
   double epsilon=1e-13;
   Print("---- créons un très petit nombre");
   Print("epsilon=",epsilon); // recevrons  epsilon=1E-013
//--- maintenant nous décomptons l'epsilon du nombre b et nous déduisons de nouveau la
   b=b-epsilon;
//--- déduisons par deux moyens
   Print("---- après la soustraction d'epsilon de la variable b");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- déduit au journal "Experts"
// Print(b)=199.9999999999999  (maintenant la valeur b après la soustraction de l'epsi
// Print(DoubleToString(b,16))=199.9999999999998578
//    (maintenant la valeur b après la soustraction de l'epsilon ne peut pas être arro
  }
```

Voir aussi

DoubleToString, StringFormat

© 2000-2015, MetaQuotes Software Corp.

PrintFormat

Formate et imprime l'ensemble des symboles et des valeurs dans un fichier-journal conformément au format donné.

```
void  PrintFormat(
   string format_string,    // ligne de format
   ...                      // valeurs des types simples
   );
```

Paramètres

format_string

[in] La ligne du format comprend les symboles ordinaires et encore et les spécifications du format, si la ligne de format est suivie par les arguments.

...

[in] N'importe quelles valeurs divisées par les virgules.La quantité de paramètres ne peut pas excéder 64, y compris la ligne de format.

La valeur rendue

La chaîne.

Note

При работе в тестере стратегий в режиме оптимизации функция PrintFormat() не выполняется.

Le nombre, l'ordre et le type de paramètres doit correspondre exactement à la composition des spécificateurs, sinon le résultat de l'impression est indéfini. Au lieu de la fonction PrintFormat() on peut utiliser la fonction printf().

Si la ligne du format est suivie encore par les paramètres, cette ligne doit contenir les spécifications du format définissant le format de la sortie de ces paramètres. La spécification du format commence toujours par le symbole du signe du pour-cent (%).

La ligne du format est lue de gauche à droite. Quand il y a une première spécification du format (si elle est), la valeur du premier paramètre après la ligne du format est transformée et est déduit selon la spécification donnée. La deuxième spécification du format appelle la transformation et la conclusion du deuxième paramètre etc, jusqu'à la fin de la ligne du format.

La spécification du format a la forme suivante:

%[flags][width][.precision][{h | l | ll | I32 | I64}]type

Chaque champ de la spécification de format est ou le symbole simple, ou le nombre désignant l'option ordinaire de format. La spécification la plus simple du format contient seulement le signe du pour-cent (%) et le symbole définissant le type du paramètre déduit (par exemple %s). S'il faut déduire dans la ligne de format le symbole le signe du pour-cent, il est nécessaire d'utiliser la spécification de format %%.

flags

Drapeau	Description	Comportement par défaut

© 2000-2015, MetaQuotes Software Corp.

- (moins)	L'alignement selon le bout gauche dans la limite de la largeur donnée	L'alignement selon le bord droit
+ (plus)	La sortie du signe + ou - pour les valeurs des types des signes	Le signe est déduit seulement si la valeur est négative
0 (zéro)	Devant la valeur déduite on ajoute les zéros dans la limite de la largeur donnée. Si on a indiqué le drapeau **0** avec le format entier (**i**, **u**, **x**, **X**, **o**, **d**) et la spécification de l'exactitude est spécifiée (par exemple, %04.d), 0 est ignoré.	Rien n'est pas inséré
espace	Devant la valeur déduite on met l'espace, si la valeur est de signe et positive	Les espaces ne sont pas insérés
#	S'il est utilisé en commun avec le format **o**, **x** ou **X**, devant la valeur déduite est ajouté 0, 0x ou 0X conformément.	Rien n'est pas inséré
	S'il est utilisé en commun avec le format **e**, **E**, **a** ou **A**, cette valeur est déduite toujours avec le point décimal.	Le point décimal est déduit seulement s'il y a une partie non zéro fractionnaire.
	S'il est utilisé en commun avec le format **g** ou **G**, le drapeau définit la présence du point décimal dans la valeur déduite et empêche l'interruption des zéros principaux Le drapeau # est ignoré à l'utilisation commune avec les formats **c**, **d**, **i**, **u**, **s**.	Le point décimal est déduit seulement s'il y a une partie non zéro fractionnaire. Les zéros principaux sont coupés

width

Le nombre non négatif décimal, qui spécifie le nombre minimal des symboles déduits de la valeur formaté. Si la quantité de symboles déduits est moins de la largeur indiquée, on ajoute la quantité correspondante des espaces à gauche ou à droite suivant l'alignement (le drapeau -). En présence du drapeau le zéro (0), devant la valeur déduite est ajouté la quantité correspondante de zéros. Si le nombre de symboles déduits plus de largeur spécifiée, la valeur déduite n'est jamais tronquée

© 2000-2015, MetaQuotes Software Corp.

Si à titre de la largeur on indique l'astérisque (*), dans la liste des paramètres transmis à la place correspondante doit être une valeur du type int, qui sera utilisé pour l'indication de la largeur de la valeur déduite.

precision

Le nombre non négatif décimal, qui définit l'exactitude de lasortie – la quantité de chiffres après le point décimal. À la différence de la spécification de la largeur, la spécification de l'exactitude peut couper la partie de la signification fractionnaire avec l'arrondissement ou sans arrondissement

Pour des différents types (**type**)de format la spécification de l'exactitude est appliquée différemment.

Types	Description	Comportement par défaut
a, A	La spécification de l'exactitude indique la quantité de signes après le point décimal	L'exactitude par défaut – 6.
c, C	N'est pas appliqué	
d, i, u, o, x, X	Indique le nombre minimal des chiffres déduits. Si le nombre de chiffres dans un paramètre correspondant est moins que exactitude indiquée les zéros sont ajoutés à la valeur déduite. La valeur déduite ne se coupe pas, si la quantité de chiffres déduits plus d'exactitude indiquée	L'exactitude par défaut – 1.
e, E, f	On indique le nombre de chiffres déduits après le point décimal. Le dernier chiffre déduit s'arrondit	L'exactitude par défaut – 6. Si on indique l'exactitude 0 ou la partie fractionnaire manque, le point décimal n'est pas déduit.
g, G	On indique le nombre maximum des chiffres signifiants	On déduit 6 chiffres signifiants.
s, S	On indique la quantité de symboles déduits de la ligne. Si la longueur la ligne excède la valeur de l'exactitude, la ligne est tronquée sur la sortie	On déduit toute la ligne

h | l | ll | I32 | I64

© 2000-2015, MetaQuotes Software Corp.

Les spécifications des grandeurs des données transmises à titre du paramètre.

Type du para-mètre	Préfixe utilisé	Spécificateur commun du type
int	l (minuscule L)	d, i, o, x, or X
uint	l (minuscule L)	o, u, x, or X
long	ll (deux minuscules L)	d, i, o, x, or X
short	h	d, i, o, x, or X
ushort	h	o, u, x, or X
int	I32	d, i, o, x, or X
uint	I32	o, u, x, or X
long	I64	d, i, o, x, or X
ulong	I64	o, u, x, or X

type

Le spécificateur du type est le seul champ obligatoire pour la sortie formatée.

Symbole	Type	Format de sortie
c	int	Le symbole du type short (Unicode)
C	int	Le symbole du type char (ANSI)
d	int	L'entier de signe décimal
i	int	L'entier de signe décimal
o	int	Le nombre entier octal sans signe
u	int	Le nombre entier décimal sans signe
x	int	Le nombre entier hexadecimal sans signe avec l'utilisation "abcdef"
X	int	Le nombre entier hexadecimal sans signe avec l'utilisation "ABCDEF"
e	double	La valeur réelle au format [-]d.dddd **e** [sign]ddd, où d -

© 2000-2015, MetaQuotes Software Corp.

		un chiffre décimal, dddd - un ou plus des chiffres décimaux, ddd -le nombre à trois chiffres, qui détermine la grandeur de l'exposant, sign - le signe plus ou le moins
E	double	Identiquement au format e, excepté que le signe d'exposant est déduit par la majuscule (**E** au lieu d'e)
f	double	La valeur réelle au format [-]dddd.dddd, où dddd - un ou plus de chiffres décimaux. La quantité de signes déduits devant le point décimal dépend de la valeur de la signification du nombre. Le nombre de signes après le point décimal dépend de l'exactitude demandée.
g	double	La valeur réelle, déduite au format **f** ou **e**, en fonction de celui-là quelle sortie sera plus compacte.
G	double	La valeur réelle, déduite au format **f** ou **E**, en fonction de celui-là quelle sortie sera plus compacte.
a	double	La valeur réelle au format [-] 0xh.hhhh **p**±dd, où h.hhhh - la mantisse dans la forme de chiffres hexadécimaux avec l'utilisation"abcdef", dd - un ou plus de chiffres de l'exposant. Le nombre de signes après la virgule est définie par la spécification de l'exactitude
A	double	La valeur réelle au format [-] 0xh.hhhh **P**±dd, où h.hhhh - la mantisse dans la forme de chiffres hexadécimaux avec l'utilisation "ABCDEF", dd - un ou plus de chiffres de l'exposant. Le nombre de signes après la virgule est

© 2000-2015, MetaQuotes Software Corp.

		définie par la spécification de l'exactitude
s	string	La sortie de la ligne

Au lieu de la fonction PrintFormat() on peut utiliser la fonction printf().

Exemple:

```
void OnStart()
  {
//--- имя торгового сервера
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- номер торговго счета
   int login=(int)AccountInfoInteger(ACCOUNT_LOGIN);
//--- вывод значения long
   long leverage=AccountInfoInteger(ACCOUNT_LEVERAGE);
   PrintFormat("%s %d: плечо = 1:%I64d",
               server,login,leverage);
//--- валюта депозита
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- вывод значения double с 2 цифрами после десятичной точки
   double equity=AccountInfoDouble(ACCOUNT_EQUITY);
   PrintFormat("%s %d: размер собственных средств на счете = %.2f %s",
               server,login,equity,currency);
//--- вывод значения double с обязательным выводом знака +/-
   double profit=AccountInfoDouble(ACCOUNT_PROFIT);
   PrintFormat("%s %d: текущий результат по открытым позициям = %+.2f %s",
               server,login,profit,currency);
//--- вывод значения double с переменным количеством цифр после десятичной точки
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   string format_string=StringFormat("%%s: значение одного пункта  = %%.%df",_Digits);
   PrintFormat(format_string,_Symbol,point_value);
//--- вывод значения int
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   PrintFormat("%s: текущий спред в пунктах = %d ",
               _Symbol,spread);
//--- вывод значения double в научном формате с плавающей запятой и точностью 17 знака
   PrintFormat("DBL_MAX = %.17e",DBL_MAX);
//--- вывод значения double в научном формате с плавающей запятой и точностью 17 знака
   PrintFormat("EMPTY_VALUE = %.17e",EMPTY_VALUE);
//--- вывод через PrintFormat() с точностью по умолчанию
   PrintFormat("PrintFormat(EMPTY_VALUE) = %e",EMPTY_VALUE);
//--- простой вывод через Print()
   Print("Print(EMPTY_VALUE) = ",EMPTY_VALUE);
/* результат выполнения
   MetaQuotes-Demo 1889998: плечо = 1:100
   MetaQuotes-Demo 1889998: размер собственных средств на счете = 22139.86 USD
   MetaQuotes-Demo 1889998: текущий результат по открытым позициям = +174.00 USD
   EURUSD: значение одного пункта  = 0.00001
   EURUSD: текущий спред в пунктах = 12
   DBL_MAX = 1.79769313486231570e+308
   EMPTY_VALUE = 1.79769313486231570e+308
   PrintFormat(EMPTY_VALUE) = 1.797693e+308
   Print(EMPTY_VALUE) = 1.797693134862316e+308
*/
  }
```

© 2000-2015, MetaQuotes Software Corp.

Voir aussi

StringFormat, DoubleToString, Les types matériels (double, float)

© 2000-2015, MetaQuotes Software Corp.

ResetLastError

Met la valeur de la variable prédéterminée _LastError dans le zéro.

```
void  ResetLastError();
```

La valeur rendue

Il n'y a pas de valeur rendue.

Note

Il est nécessaire de noter que la fonction GetLastError() ne remet pas au zéro la valeur _LastError. D'habitude l'appel de la fonction est produit avant l'appel de la fonction, après laquelle l'apparition de l'erreurest vérifiée.

© 2000-2015, MetaQuotes Software Corp.

Enter topic text here.

ResourceCreate

Crée la ressource de l'image à la base de l'ensemble des données. Il y a 2 variantes de la fonction:

La création de la ressource à la base du fichier

```
bool  ResourceCreate(
   const string      resource_name,       // le nom de la ressource
   const string      path                 // le chemin relatif au fichier
   );
```

La création de la ressource à la base du tableau des pixels

```
bool  ResourceCreate(
   const string      resource_name,       // le nom de la ressource
   const uint&       data[],              // un ensemble de données dans un tableau
   uint              img_width,           // la largeur de l'image - la ressource crée
   uint              img_height,          // la hauteur de l'image - la ressource crée
   uint              data_xoffset,        // le déplacement de l'angle gauche supérieu
   uint              data_yoffset,        // le déplacement de l'angle gauche supérieu
   uint              data_width,          // la largeur totale de l'image basée sur u
   ENUM_COLOR_FORMAT color_format         //la largeur totale de l'image basée sur un
   );
```

Paramètres

resource_name

[in] Nom de la ressource.

data[][]

[in] Un tableau unidimensionnel ou bidimensionnel pour créer une image complète.

img_width

[in] La largeur du domaine rectangulaire de l'image en pixels pour la mise dans la ressource en forme de l'image. Plus de valeur ne peut pas être*data_width*.

img_height

[in] La hauteur du domaine rectangulaire de l'image en pixels pour la mise dans la ressource en forme de l'image.

data_xoffset

[in] Le déplacement en pixels du domaine rectangulaire de l'image à l'horizontale à droite.

data_yoffset

[in] Le déplacement en pixels du domaine rectangulaire de l'image selon la verticale en bas.

data_width

[in] Il est nécessaire seulement pour les tableaux unidimentionnels et signifie la largeur complète de l'image créée de l'ensemble de données. Si *data_width*=0, il est supposé égal à *img_width*. Pour les tableaux bidimensionnels ce paramètre est ignoré et est accepté comme égal à la deuxième dimension du tableau *data[]*.

color_format

[in] Le moyen du traitement de la couleur de l'énumération ENUM_COLOR_FORMAT.

La valeur rendue

© 2000-2015, MetaQuotes Software Corp.

true – en cas du succès, autrement false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError(). Les erreurs possibles:

- 4015 – ERR_RESOURCE_NAME_DUPLICATED (la coïncidence des noms des ressourcesstatiqueet dynamique),
- 4016 – ERR_RESOURCE_NOT_FOUND (la ressource n'est pas trouvée),
- 4017 – ERR_RESOURCE_UNSUPPOTED_TYPE (le type de la ressource n'est pas soutenu),
- 4018 – ERR_RESOURCE_NAME_IS_TOO_LONG (le nom de la ressource est trop long).

Note

Si la deuxième variante de la fonction est appelée pour créer la même ressource avec de différents paramètres de la largeur, de la hauteur et du décalage, une nouvelle ressource n'est pas créée de nouveau, mais on met à jour une ressource existante tout simplement.

La première variante de la fonction permet de charger les images et les sons des fichiers, la deuxième variante est destinée seulement à la création dynamique des images.

Les images doivent être au format BMP avec une profondeur de couleur de 24 ou 32 bits, le son peut être seulement au format WAV. Le montant du fichier de la ressource ne peut pas être plus de 16 Mb.

ENUM_COLOR_FORMAT

Identificateur	La description
COLOR_FORMAT_XRGB_NOALPHA	Le composant du canal alpha est ignoré
COLOR_FORMAT_ARGB_RAW	Les composants de la couleur ne sont pas traités par le terminal (ils doivent être correctement spécifiés par l'utilisateur)
COLOR_FORMAT_ARGB_NORMALIZE	Les composants de la couleur sont traités par le terminal

Voir aussi

Les ressources, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2015, MetaQuotes Software Corp.

ResourceFree

Supprime la ressource dynamiquement créée(libère la mémoire occupée par la ressource).

```
bool  ResourceFree(
   const string  resource_name      // le nom de la ressource
   );
```

Paramètres

resource_name

[in] Le nom dela ressource, doit commencer par "::".

La valeur rendue

true - en cas du succès, autrement false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

La fonction ResourceFree () permet au développeur du programme mql5 de maîtriser la consommation de la mémoire au travail actif avec les ressources. Les objets graphiques, attachés à la ressource supprimée de la mémoire, s'afficheront correctement aussi après son suppression. Mais les objets graphiques créés de nouveau (OBJ_BITMAP et OBJ_BITMAP_LABEL) ne pourront pas utiliser la ressource supprimée.

La fonction supprime seulement les ressources dynamiques créées par ce programme.

Voir aussi

Les ressources, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2015, MetaQuotes Software Corp.

ResourceReadImage

Lit les données de la ressource graphique, créé par la fonction ResourceCreate() ou sauvegardée dans le fichier EX5 à la compilation.

```
bool  ResourceReadImage(
   const string      resource_name,       // le nom de la ressource graphique pour la
   uint&             data[],              // le tableau pour la réception des données
   uint&             width,               // pour la réception de la largeur de l'imag
   uint&             height,              // pour la réception de la hauteur de l'imag
   );
```

Paramètres

resource_name

[in] Le nom de la ressource graphique contenant l'image. Pour l'accès aux ressources personnelles on indique court "::resourcename". S'il est nécessaire de charger la ressource du fichier EX5 compilé, il est nécessaire le nom dans l'aspect complet avec l'indication de la voie relativement le dossier MQL5, le nom du fichier et le nom de la ressource- "path\\filename.ex5::resourcename".

data[][]

[in] Un tableau unidimentionnel ou à deux dimensions pour la réception des données de la ressource graphique.

img_width

[out] La largeur de l'image de la ressource graphique en pixels.

img_height

[out] La hateur de l'image de la ressource graphique en pixels.

La valeur rendue

true - en cas du succès, autrement false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

Si à la base du tableau*data[]* il est nécessairede créer après la ressource graphique, il faut utiliser le format de la couleur COLOR_FORMAT_ARGB_NORMALIZE ou COLOR_FORMAT_XRGB_NOALPHA.

Si le tableau*data[]* est bidimensionnel et sa deuxième dimension est moins que la taille X(width) de la ressource graphique, la fonction ResourceReadImage() rendra false et la lecture ne sera pas produite. Mais en cela, si la ressource existe, les tailles actuelles de l'image reviennent aux paramètres width et height. Cela permettra de faire encore une tentative de la réception des données de la ressource.

Voir aussi

Les ressources, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2015, MetaQuotes Software Corp.

ResourceSave

Sauvegarde la ressource au fichier spécifié.

```
bool  ResourceSave(
   const string  resource_name      // le nom de la ressource
   const string  file_name          // le nom du fichier
   );
```

Paramètres

resource_name

[in] Le nom de la ressource doit commencer par "::".

file_name

[in] Le nom du fichier relativement MQL5\Files.

La valeur rendue

true - si en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

La fonction réenregistre toujours le fichier et en cas de nécessité crée tous les sous-répertoires intermédiaires dans le nom du fichier en cas de leur absence.

Voir aussi

Les ressources, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2015, MetaQuotes Software Corp.

SetUserError

Établit la variable prédéterminée _LastError à la valeur égale à ERR_USER_ERROR_FIRST + user_error

```
void  SetUserError(
   ushort user_error,    // numéro de l'erreur
   );
```

Paramètres

user_error

[in] Le numero de l'erreur , établi par l'utilisateur.

La valeur rendue

Il n'y a pas de valeur rendue.

Note

Après qu'une erreur a été mise à l'aide de la fonction SetUserError(user_error), la fonction GetLastError() rendra la valeur, égal à ERR_USER_ERROR_FIRST + user_error.

Exemple:

```
void OnStart()
  {
//--- définissons le numéro de l'erreur 65537=(ERR_USER_ERROR_FIRST +1)
   SetUserError(1);
//--- recevrons le code de la dernière erreur
   Print("GetLastError = ",GetLastError());
/*
   Résultat
   GetLastError = 65537
*/
  }
```

© 2000-2015, MetaQuotes Software Corp.

SendFTP

Envoie le fichier à l'adresse indiquée dans la fenêtre des réglages du signet "la publication".

```
bool  SendFTP(
   string  filename,          // fichier pour l'envoi à
   string  ftp_path=NULL      // le chemin vers le déchargement sur le serveur ftp
   );
```

Paramètres

filename

[in] Le nom du fichier envoyé.

ftp_path=NULL

[in] Le répertoire FTP. Si le répertoire n'est pas indiqué, on utilise le répertoire décrit dans les configurations.

La valeur rendue

En cas de l'échec rend false.

Note

Le fichier envoyé doit se trouver dans le dossier *le répertoire_du terminal\MQL5\files* ou dans ses sous-dossiers. L'envoi n'est pas produit, si dans les configurations on n'indique pas l'adresse FTP et/ou le mot de passe de l'accès.

При работе в тестере стратегий функция SendFTP() не выполняется.

© 2000-2015, MetaQuotes Software Corp.

SendNotification

Envoie la notification aux terminaux mobiles dont ID MetaQuotes sont spécifiés dans la fenêtre des options sur l'onglet "Notifications"

```
bool  SendNotification(
   string  text          // le texte du message
   );
```

Paramètres

text

[in] Le texte du message dans la notification. La longueur du message doit être pas plus de 255 symboles.

La valeur rendue

true) l'envoi réussi de la notification à partir du terminal, en cas de l'échec rend false. A la vérification après un mauvais envoi de la notificationGetLastError() une des erreurs suivante peut être donnée:

- 4515 - ERR_NOTIFICATION_SEND_FAILED,
- 4516 - ERR_NOTIFICATION_WRONG_PARAMETER,
- 4517 - ERR_NOTIFICATION_WRONG_SETTINGS,
- 4518 - ERR_NOTIFICATION_TOO_FREQUENT.

Note

Pour la fonction SendNotification() il y a des limitations strictes selon l'utilisation: pas plus de 2 appels par seconde et pas plus de 10 appels par minute. Le contrôle de la fréquence de l'utilisation est fait dynamiquement, et la fonction peut être bloquée à la violation.

При работе в тестере стратегий функция SendNotification() не выполняется.

© 2000-2015, MetaQuotes Software Corp.

SendMail

Envoie le message électronique à l'adresse indiquée dans la fenêtre des réglages du signet "la Poste"

```
bool  SendMail(
   string  subject,        // titre
   string  some_text       // texte du message
   );
```

Paramètres

subject

[in] Le titre du message.

some_text

[in] Le corps du message.

La valeur rendue

true - si le message est mise au tour sur l'envoi, autrement rend false.

Note

L'envoi peut être interdit dans les configurations, ou peut être on n'a pas indiqué l'adresse de courrier électronique. Pour recevoir l'information sur l'erreur, il est nécessaire d' appeler la fonction GetLastError().

При работе в тестере стратегий функция SendMail() не выполняется.

© 2000-2015, MetaQuotes Software Corp.

Sleep

Retient l'exécution de l'expert courant ou le script sur l'intervalle défini.

```
void  Sleep(
   int  milliseconds      // intervalle
   );
```

Paramètres

milliseconds

[in] L'interligne du retard des millisecondes.

La valeur rendue

Il n'y a pas de valeur rendue.

Note

On ne peut pas appeler la fonction Sleep() des indicateurs d'utilisateur, puisque les indicateurs sont accomplis dans le thread d'interface et ne doivent pas le ralentir. La vérification de l'état du drapeau du stoppage de l'expert de chaque 0.1 secondes est inseré dans la fonction.

© 2000-2015, MetaQuotes Software Corp.

TerminalClose

Envoie au terminal l'ordre sur l'achèvement du travail.

```
bool  TerminalClose(
   int ret_code      // code de retour du terminal de client
   );
```

Paramètres

ret_code

[in] Le code de retour rendu par le procès du terminal de client à l'achèvement du travail.

La valeur rendue

Rend true en cas du succès, autrement false.

Note

La fonction TerminalClose() ne produit pas l'arrêt immédiat du travail du terminal, elle envoie simplement au terminal la commande sur l'achèvement.

Dans le code du conseiller qui a appelé TerminalClose(), on doit faire toutes les préparations pour l'achèvement immédiat du travail (par exemple, tous les fichiers auparavant ouverts doivent être fermés). Après l'appel de cette fonction va l'opérateur return.

Le paramètre *ret_code* permet d'indiquer le code de retour nécessaire pour l'analyse des raisons de la cessation du travail de programme du terminal à son lancement de la ligne d'instruction.

Exemple:

```
//--- input parameters
input int  tiks_before=500; // quantité de ticks avant l'achèvement
input int  pips_to_go=15;   // distance aux pips
input int  seconds_st=50;   // combien de secondes donnons-nous à l'expert
//--- globals
datetime   launch_time;
int        tick_counter=0;
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(__FUNCTION__," reason code = ",reason);
   Comment("");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   static double first_bid=0.0;
```

© 2000-2015, MetaQuotes Software Corp.

```
   MqlTick       tick;
   double        distance;
//---
   SymbolInfoTick(_Symbol,tick);
   tick_counter++;
   if(first_bid==0.0)
     {
      launch_time=tick.time;
      first_bid=tick.bid;
      Print("first_bid = ",first_bid);
      return;
     }
//--- marche du prix dans les points
   distance=(tick.bid-first_bid)/_Point;
//--- montrons le message pour surveiller au travail de conseiller
   string comm="Après le moment du lancement :\r\n\x25CF a passé des secondes: "+
               IntegerToString(tick.time-launch_time)+" ;"+
               "\r\n\x25CF ticks reçues: "+(string)tick_counter+" ;"+
               "\r\n\x25CF prix a passé dans les points: "+StringFormat("%G",distance);
   Comment(comm);
//--- section de la vérification des conditions de la clôture du terminal
   if(tick_counter>=tiks_before)
      TerminalClose(0);    // sortie selon le compteur des ticks
   if(distance>pips_to_go)
      TerminalClose(1);    // ont passé en haut aux pips_to_go pips
   if(distance<-pips_to_go)
      TerminalClose(-1);   // ont passé en bas aux pips_to_go pips
   if(tick.time-launch_time>seconds_st)
      TerminalClose(100);  // achèvement du travail selon le timeout
//---
  }
```

Voir aussi

L'exécution des programmes, Les erreurs de l'exécution, Les raisons de la déinitialisation

© 2000-2015, MetaQuotes Software Corp.

Enter topic text here.

TesterStatistics

Rend la valeur du paramètre indiqué statistique, calculée d'après les résultats du test

```
double  TesterStatistics(
   ENUM_STATISTICS statistic_id      // l'identificateur
   );
```

Paramètres

statistic_id

[in] L'identificateur du paramètre statistique de l'énumérationENUM_STATISTICS.

La valeur rendue

La valeur du paramètre statistique des résultats du test.

Note

La fonction peut être appelée au sein d' OnTester () ou OnDeinit () dans le testeur. Dans d'autres cas, le résultat est indéfini.

© 2000-2015, MetaQuotes Software Corp.

TesterWithdrawal

La fonction spéciale pour l'émulation des opérations du main levée des moyens pendant le test. Peut être utilisé dans certains systèmes de gestion du capital.

```
bool  TesterWithdrawal(
   double money      // montant de la somme retirée
   );
```

Les paramètres

money

[in] Le montant des moyens de financement, dont il est nécessaire de retirer du compte (en devise du dépôt).

La valeur rendue

Rend true en cas du succès, autrement rend false.

© 2000-2015, MetaQuotes Software Corp.

WebRequest

Отправляет HTTP-запрос на указанный сервер. Существует два варианта функций:

1. **Для отправки простых запросов** вида "ключ=значение" с использованием заголовка Content-Type: application/x-www-form-urlencoded.

```
int  WebRequest(
   const string      method,           // метод HTTP
   const string      url,              // url-адрес
   const string      cookie,           // cookie
   const string      referer,          // referer
   int               timeout,          // таймаут
   const char        &data[],          // массив тела HTTP-сообщения
   int               data_size,        // размер массива data[] в байтах
   char              &result[],        // массив с данными ответа сервера
   string            &result_headers   // заголовки ответа сервера
   );
```

2. **Для отправки запросов произвольного типа** с указанием собственного набора заголовков для более гибкого взаимодействия с различными Web-сервисами.

```
int  WebRequest(
   const string      method,           // метод HTTP
   const string      url,              // url-адрес
   const string      headers,          // заголовки
   int               timeout,          // таймаут
   const char        &data[],          // массив тела HTTP-сообщения
   char              &result[],        // массив с данными ответа сервера
   string            &result_headers   // заголовки ответа сервера
   );
```

Параметры

method

[in] Метод HTTP.

url

[in] URL-адрес.

headers

[in] Заголовки запроса вида "ключ: значение", разделенные переносом строки "\r\n".

cookie

[in] Значение Cookie.

referer

[in] Значение заголовка Referer HTTP-запроса.

timeout

[in] Таймаут в миллисекундах.

© 2000-2015, MetaQuotes Software Corp.

data[]

[in] Массив данных тела HTTP-сообщения.

data_size

[in] Размер массива data[].

result[]

[out] Массив с данными ответа сервера.

result_headers

[out] Заголовки ответа сервера.

Возвращаемое значение

Код ответа HTTP-сервера, либо -1 в случае ошибки.

Примечание

Для использования функции WebRequest() следует добавить адреса серверов в список разрешенных URL во вкладке "Советники" окна "Настройки". Порт сервера выбирается автоматически на основе указанного протокола - 80 для "http://" и 443 для "https://".

Функция WebRequest() является синхронной, это означает, что она приостанавливает выполнение программы и ждет ответа от запрашиваемого сервера. Так как задержки при получении ответа на отправленный запрос могут быть большими, то функция запрещена для вызовов из индикаторов, поскольку индикаторы работают в собственных потоках, сгруппированных по символам. Задержка выполнения индикатора на одном из графиков символа может привести к остановке обновления всех графиков по данному символу.

Функцию можно вызывать только из экспертов и скриптов, так как они работают в собственном потоке выполнения. При вызове из индикатора GetLastError() вернет ошибку 4014 - "Функция не разрешена".

При работе в тестере стратегий функция WebRequest() не выполняется.

Пример использования 1-го варианта функции WebRequest():

```
void OnStart()
  {
   string cookie=NULL,headers;
   char post[],result[];
   int res;
//--- для работы с сервером необходимо добавить URL "https://www.google.com/finance"
//--- в список разрешенных URL (Главное меню->Сервис->Настройки, вкладка "Советники");
   string google_url="https://www.google.com/finance";
//--- обнуляем код последней ошибки
   ResetLastError();
//--- загрузка html-страницы с Google Finance
   int timeout=5000; //--- timeout менее 1000 (1 сек.) недостаточен при низкой скорост
   res=WebRequest("GET",google_url,cookie,NULL,timeout,post,0,result,headers);
//--- проверка ошибок
   if(res==-1)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      Print("Ошибка в WebRequest. Код ошибки  =",GetLastError());
      //--- возможно URL отсутствует в списке, выводим сообщение о необходимости его д
      MessageBox("Необходимо добавить адрес '"+google_url+"' в список разрешенных URL
     }
   else
     {
      //--- успешная загрузка
      PrintFormat("Файл успешно загружен, Размер файла =%d байт.",ArraySize(result));
      //--- сохраняем данные в файл
      int filehandle=FileOpen("GoogleFinance.htm",FILE_WRITE|FILE_BIN);
      //--- проверка ошибки
      if(filehandle!=INVALID_HANDLE)
        {
         //--- сохраняем содержимое массива result[] в файл
         FileWriteArray(filehandle,result,0,ArraySize(result));
         //--- закрываем файл
         FileClose(filehandle);
        }
      else Print("Ошибка в FileOpen. Код ошибки =",GetLastError());
     }
  }
```

Пример использования 2-го варианта функции WebRequest():

```
#property link      "http://www.mql5.com"
#property version   "1.00"
#property strict
#property script_show_inputs
#property description "Пример скрипта, который публикует сообщение "
#property description "пользователя в ленте на mql5.com"

input string InpLogin    ="";              //Ваш аккаунт в MQL5.com
input string InpPassword="";               //Пароль для вашего аккаунта
input string InpFileName="EURUSDM5.png"; //Картинка в папке MQL5/Files/
input string InpFileType="image/png";     //Правильный mime type картинки
//+------------------------------------------------------------------+
//| Публикация сообщения с картинкой в ленте mql5.com                |
//+------------------------------------------------------------------+
bool PostToNewsFeed(string login,string password,string text,string filename,string fi
  {
   int     res;     // для помещения результата выполнения операций
   char    data[];  // массив с данными для отправки POST-запросов
   char    file[];  // сюда прочитаем картинку
   string str="Login="+login+"&Password="+password;
   string auth,sep="-------Jyecslin9mp8RdKV"; // разделитель данных формата multipart
//--- имеется файл - пробуем прочитать его
   if(filename!=NULL && filename!="")
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      res=FileOpen(filename,FILE_READ|FILE_BIN);
      if(res<0)
        {
         Print("Ошибка открытия файла \""+filename+"\"");
         return(false);
        }
      //--- читаем данные файла
      if(FileReadArray(res,file)!=FileSize(res))
        {
         FileClose(res);
         Print("Ошибка чтения файла \""+filename+"\"");
         return(false);
        }
      //---
      FileClose(res);
     }
//--- сформируем тело POST запроса на авторизацию
   ArrayResize(data,StringToCharArray(str,data,0,WHOLE_ARRAY,CP_UTF8)-1);
//--- сбросим код ошибки
   ResetLastError();
//--- выполняем запрос на автиризацию
   res=WebRequest("POST","https://www.mql5.com/ru/auth_login",NULL,0,data,data,str);
//--- если авторизация не удалась
   if(res!=200)
     {
      Print("Ошибка авторизации #"+(string)res+", LastError="+(string)GetLastError());
      return(false);
     }
//--- вычитаем из заголовка ответа сервера авторизационную куку
   res=StringFind(str,"Set-Cookie: auth=");
//--- если кука не найдена - сообщим об ошибке
   if(res<0)
     {
      Print("Ошибка, данные авторизации не найдены в ответе сервера (проверте логин/па
      return(false);
     }
//--- запомним авторизационные данные и сформируем заголовок для последующих запросов
   auth=StringSubstr(str,res+12);
   auth="Cookie: "+StringSubstr(auth,0,StringFind(auth,";")+1)+"\r\n";
//--- если имеется файл данных - отправляем его на сервер
   if(ArraySize(file)!=0)
     {
      //--- сформируем тело запроса
      str="--"+sep+"\r\n";
      str+="Content-Disposition: form-data; name=\"attachedFile_imagesLoader\"; filena
      str+="Content-Type: "+filetype+"\r\n\r\n";
      res =StringToCharArray(str,data);
      res+=ArrayCopy(data,file,res-1,0);
      res+=StringToCharArray("\r\n--"+sep+"--\r\n",data,res-1);
```

© 2000-2015, MetaQuotes Software Corp.

```
      ArrayResize(data,ArraySize(data)-1);
      //--- сформируем заголовок запроса
      str=auth+"Content-Type: multipart/form-data; boundary="+sep+"\r\n";
      //--- сбросим код ошибки
      ResetLastError();
      //--- выполняем запрос на передачу файла изображения на сервер
      res=WebRequest("POST","https://www.mql5.com/upload_file",str,0,data,data,str);
      //--- проверим результат запроса
      if(res!=200)
        {
         Print("Ошибка передачи файла на сервер #"+(string)res+", LastError="+(string)
         return(false);
        }
      //--- получим ссылку на картинку, которую загрузили на сервер
      str=CharArrayToString(data);
      if(StringFind(str,"{\"Url\":\"")==0)
        {
         res      =StringFind(str,"\"",8);
         filename=StringSubstr(str,8,res-8);
         //--- при ошибке закачки файла, вернётся пустая ссылка
         if(filename=="")
           {
            Print("Передача файла на сервер не удалась");
            return(false);
           }
        }
     }
//--- сформируем тело запроса публикации сообщения на сервере
   str ="--"+sep+"\r\n";
   str+="Content-Disposition: form-data; name=\"content\"\r\n\r\n";
   str+=text+"\r\n";
//--- на каких языках сайта mql5.com будет доступна публикация
   str+="--"+sep+"\r\n";
   str+="Content-Disposition: form-data; name=\"AllLanguages\"\r\n\r\n";
   str+="on\r\n";
//--- если картинка была загружена на сервер - передадим ссылку на неё
   if(ArraySize(file)!=0)
     {
      str+="--"+sep+"\r\n";
      str+="Content-Disposition: form-data; name=\"attachedImage_0\"\r\n\r\n";
      str+=filename+"\r\n";
     }
//--- завершающая строка multipart-запроса
   str+="--"+sep+"--\r\n";
//--- собираем тело POST-запроса в одну строку
   StringToCharArray(str,data,0,WHOLE_ARRAY,CP_UTF8);
   ArrayResize(data,ArraySize(data)-1);
//--- подготовим заголовок запроса
   str=auth+"Content-Type: multipart/form-data; boundary="+sep+"\r\n";
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- выполняем запрос на публикацию сообщение в ленте пользователя mql5.com
   res=WebRequest("POST","https://www.mql5.com/ru/users/"+login+"/wall",str,0,data,dat
//--- в случае успешной публикации вернем true
   return(res==200);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- опубликуем пост на mql5.com с картинкой, путь к которой возьмем из параметра Inp
   PostToNewsFeed(InpLogin,InpPassword,"Проверка расширенной версии функции WebRequest
                  "(Данное сообщение размещено скриптом WebRequest.mq5)",InpFileName,I
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

ZeroMemory

Remet au zéro la variable transmise selon la référence.

```
void  ZeroMemory(
   void & variable      // variable de reconstruction
   );
```

Paramètres

variable

[in] [out] La variable transmise selon le lien laquelle il faut remettre au zéro (initialisez par les valeurs zéro).

La valeur rendue

Il n'y a pas de valeur rendue.

Note

Si le paramètre de la fonction est la ligne, cet appel sera équivalent à l'indication pour elle de la valeur NULL.
Pour les types simples et leurs tableaux, ainsi que les structures/classes comprenant tels types, c'est un simple remise au zéro.
Pour les objets contenant les lignes et les tableaux dynamiques, on produit l'appel ZeroMemory() pour chaque membre.
Pour chaque tableaux non protégés par le modificateur const, la remise au zéro de tous les éléments est faite.
Pour les tableaux des objets complexes on appelle ZeroMemory() pour chaque élément.

La fonction ZeroMemory() n'est pas employée pour les classes avec les membres protégés ou l'héritage.

© 2000-2015, MetaQuotes Software Corp.

Le groupe des fonctions pour le travail avec les tableaux

Pas plus que les tableaux quadridimensionnels sont admissibles. L'indexation de chaque mesure est produite de 0 jusqu'à *la grandeur_de la dimension-1*. Dans le cas particulier de tableaux unidimentionnel des 50 éléments l'appel au premier élément aura l'air comme array[0], au dernier élément - array[49].

Fonction	Action
ArrayBsearch	Rend l'index du premier élément trouvé dans dans la première dimension du tableau
ArrayCopy	Copie un tableau à l'autre
ArrayCompare	Rend le résultat de la comparaison de deux tableux des types simples ou des structures d'utilisateur qui n'ont pas des objets complexes
ArrayFree	Libère la mémoire du tampon de n'importe quel tableau dynamique et met la dimension de la valeur zéro au 0.
ArrayGetAsSeries	Contrôle la direction de l'indexation du tableau
ArrayInitialize	Met tous les éléments du tableau numérique à une valeur
ArrayFill	Remplit le tableau numérique par la valeur indiquée
ArrayIsSeries	Vérifies - si le tableau est la série temporelle
ArrayIsDynamic	Vérifies - si le tableau est dynamique
ArrayMaximum	La recherche de l'élément avec la valeur maximal
ArrayMinimum	La recherche de l'élément avec la valeur minimal
ArrayRange	Rend le nombre d'éléments dans la dimension indiquée du tableau
ArrayResize	Met la nouvelle grandeur dans la première dimension du tableau
ArraySetAsSeries	Établit la direction de l'indexation dans le tableau
ArraySize	Rend le nombre d'éléments dans le tableau
ArraySort	Le triage des tableaux numériques par la première dimension

© 2000-2015, MetaQuotes Software Corp.

ArrayBsearch

Ищет указанное значение в отсортированном по возрастанию многомерном числовом массиве. Поиск производится по элементам первого измерения.

Pour la recherche dans le tableau du type double

```
int  ArrayBsearch(
   const double&    array[],   // tableau pour la recherche
   double           value      // ce qu'on cherche
   );
```

Pour la recherche dans le tableau du type float

```
int  ArrayBsearch(
   const float&    array[],   // tableau pour la recherche
   float           value      // ce qu'on cherche
   );
```

Pour la recherche dans le tableau du type long

```
int  ArrayBsearch(
   const long&    array[],   // tableau pour la recherche
   long           value      // ce qu'on cherche
   );
```

Pour la recherche dans le tableau du type int

```
int  ArrayBsearch(
   const int&    array[],   // tableau pour la recherche
   int           value      // ce qu'on cherche
   );
```

Pour la recherche dans le tableau du type short

```
int  ArrayBsearch(
   const short&    array[],   // tableau pour la recherche
   short           value      // ce qu'on cherche
   );
```

Pour la recherche dans le tableau du type char

```
int  ArrayBsearch(
   const char&    array[],   // tableau pour la recherche
   char           value      // ce qu'on cherche
   );
```

Paramètres

array[]

[in] Le tableau numérique pour la recherche.

value

© 2000-2015, MetaQuotes Software Corp.

[in] La valeur pour la recherche.

La valeur rendue

Rend l'index de l'élément trouvé. Si la valeur cherchée n'est pas trouvée, rend l'index de l'élément plus proche selon la valeur.

Note

La recherche binaire traite seulement les tableaux triés. Pour le triage du tableau numérique on utilise la fonction ArraySort().

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
#property description "Le script basé sur l'indicateur RSI affiche dans la fenêtre du
#property description "les données combien de fois le marché a été dans les zones surv
#property description "et de survente dans le délai spécifié.
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input int                InpMAPeriod=14;                  // La période de la moyenn
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;     // Le type du prix
input double             InpOversoldValue=30.0;           // Le niveau de la survent
input double             InpOverboughtValue=70.0;         // Le niveau du surachat
input datetime           InpDateStart=D'2012.01.01 00:00'; // La date de commencement
input datetime           InpDateFinish=D'2013.01.01 00:00'; // La date de la fin de l'
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   double rsi_buff[]; // le tableau des valeurs de l'indicateur
   int    size=0;     // la taille du tableau
//--- recevons le handle de l'indicateur RSI
   ResetLastError();
   int rsi_handle=iRSI(Symbol(),Period(),InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- on n'a pas réussi à recevoir le handle de l'indicateur
      PrintFormat("L'erreur de la réception du handle de l'indicateur. Le code de l'er
      return;
     }
//--- nous sommes dans le cycle, jusqu'à ce que l'indicateur ne calculera pas toutes s
   while(BarsCalculated(rsi_handle)==-1)
     {
      //--- sortons, si l'utilisateur a terminé forcément le travail du script
      if(IsStopped())
         return;
      //--- le retard pour que l'indicateur puisse de calculer ses valeurs
      Sleep(10);
     }
//--- copions les valeurs de l'indicateur pour la période définie
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,InpDateFinish,rsi_buff)==-1)
     {
      PrintFormat("On n' a pas réussi à copier les valeurs de l'indicateur. Le code de
      return;
     }
//--- recevons la taille du tableau
   size=ArraySize(rsi_buff);
//--- trions le tableau
   ArraySort(rsi_buff);
//--- apprenons quel pour-cent du temps le marché se trouvait dans la zone de survente
   double ovs=(double)ArrayBsearch(rsi_buff,InpOversoldValue)*100/(double)size;
//--- apprenons quel pour-cent du temps le marché se trouvait dans la zone de surachat
   double ovb=(double)(size-ArrayBsearch(rsi_buff,InpOverboughtValue))*100/(double)siz
//---formons les chaînes pour sortir les données
   string str="De "+TimeToString(InpDateStart,TIME_DATE)+" jusq'au "
              +TimeToString(InpDateFinish,TIME_DATE)+" le marché a été:";
   string str_ovb="dans la zone de surachat "+DoubleToString(ovb,2)+"% du temps";
   string str_ovs="dans la zone de survente "+DoubleToString(ovs,2)+"% du temps";
//---affichons les données sur un graphique
   CreateLabel("top",5,60,str,clrDodgerBlue);
   CreateLabel("overbought",5,35,str_ovb,clrDodgerBlue);
   CreateLabel("oversold",5,10,str_ovs,clrDodgerBlue);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- redessinons le graphique
   ChartRedraw(0);
//--- le retard
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| Affichons les commentaires dans le coin inférieur gauche du graphique
//+------------------------------------------------------------------+
void CreateLabel(const string name,const int x,const int y,
                 const string str,const color clr)
  {
//--- la création de la marque
   ObjectCreate(0,name,OBJ_LABEL,0,0,0);
//--- le rattachement de la marque à l'angle gauche inférieur
   ObjectSetInteger(0,name,OBJPROP_CORNER,CORNER_LEFT_LOWER);
//--- changeons la position du point du rattachement
   ObjectSetInteger(0,name,OBJPROP_ANCHOR,ANCHOR_LEFT_LOWER);
//--- la distance selon l'axe X du point du rattachement
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
//--- La distance selon l'axe Y du point du rattachement
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
//--- le texte de la marque
   ObjectSetString(0,name,OBJPROP_TEXT,str);
//--- la couleur du texte
   ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
//--- la taille du texte
   ObjectSetInteger(0,name,OBJPROP_FONTSIZE,12);
  }
```

© 2000-2015, MetaQuotes Software Corp.

ArrayCopy

Produit le copiage d'un tableau à l'autre.

```
int  ArrayCopy(
   void&        dst_array[],          // tableau de destination
   const void&  src_array[],          // tableau de source
   int          dst_start=0,          // de quel index nous écrivons au récepteur
   int          src_start=0,          // premier index du récepteur
   int          count=WHOLE_ARRAY     // combien d'éléments
   );
```

Paramètres

dst_array[]

[out] Le tableau-lé récepteur.

src_array[]

[in] Le tableau-la source.

dst_start=0

[in] L'index initial pour le tableau de réception. Par défaut, l'index de départ est - 0.

src_start=0

[in] L'index initial pour le tableau de source. Par défaut, l'index de départ est - 0.

count=WHOLE_ARRAY

[in] Le nombre d'éléments, qu'il faut copier. Par défaut, on copie tout le tableau (count=WHOLE_ARRAY).

La valeur rendue

Rend le nombre d'éléments copiés.

Note

Si count<0 ou count>src_size-src_start, on copie tout le reste du tableau. Les tableaux sont copiés de gauche à droite. La position de départ redéfinit correctement pour les tableaux de série en tenant compte du copiage de gauche à droite. Si le tableau est copié à lui-même, le résultat est non défini.

Si les tableaux des différents types, au copiage on produit la tentative de la transformation de chaque élément du tableau initial vers le type du tableau de réception. On peut copier le tableau de chaîne seulement au tableau de chaîne. Les tableaux des classes et des structures, contenant les objets, exigeant l'initialisation, ne sont pas copiés. On peut copier le tableau des structures seulement au tableau du même type.

Для динамических массивов с индексацией как в таймсериях производится автоматическое увеличение размера массива-приемника до количества копируемых данных (в случае, если количество копируемых данных превышает его размер). Автоматическое уменьшение размера массива-приемника не производится.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
#property description "L'indicateur sélectionne par la couleur les bougies, qui sont l
#property description "par les maximums et les minimums. La longueur de l'intervalle p
#property description "on peut spécifier des extremums à l'aide du paramètre d'entrée.
//--- les réglages de l'indicateur
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot
#property indicator_label1  "Extremums"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrLightSteelBlue,clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- une constante prédéfinie
#define INDICATOR_EMPTY_VALUE 0.0
//--- les paramètres d'entrée
input int InpNum=4; // La longueur du semi-intervalle
//--- les tampons d'indicateur
double ExtOpen[];
double ExtHigh[];
double ExtLow[];
double ExtClose[];
double ExtColor[];
//--- les variables globales
int    ExtStart=0; // l'indice de la première bougie, qui n'est pas l'extremum
int    ExtCount=0; // le nombre de bougies non extremums dans l'intervalle donné
//+------------------------------------------------------------------+
//| Le coloriage des bougies non extremums                           |
//+------------------------------------------------------------------+
void FillCandles(const double &open[],const double &high[],
                 const double &low[],const double &close[])
  {
//--- colorons les bougies
   ArrayCopy(ExtOpen,open,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtHigh,high,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtLow,low,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtClose,close,ExtStart,ExtStart,ExtCount);
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ExtOpen);
   SetIndexBuffer(1,ExtHigh);
   SetIndexBuffer(2,ExtLow);
   SetIndexBuffer(3,ExtClose);
   SetIndexBuffer(4,ExtColor,INDICATOR_COLOR_INDEX);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- spécifions la valeur qui ne peut être affichée
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- spécifions les noms des tampons d'indicateurs pour l'affichage dans la fenêtre d
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---établissons la direction directe de l'indexation dans les series temporelles
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- la variable du début pour calculer les barres
   int start=prev_calculated;
//--- on ne fait pas le calcul pour les premières InpNum*2 barres
   if(start==0)
     {
      start+=InpNum*2;
      ExtStart=0;
      ExtCount=0;
     }
//---si la barre a été formé, vérifions le prochain extremum potentiel
   if(rates_total-start==1)
      start--;
//--- l'indice de la barre, lequel on va vérifier à l'extremum
   int ext;
//--- le cycle du calcule des valeurs de l'indicateur
   for(int i=start;i<rates_total-1;i++)
     {
      //--- initialement à la i-ème barre sans rendu
      ExtOpen[i]=0;
      ExtHigh[i]=0;
      ExtLow[i]=0;
      ExtClose[i]=0;
      //--- l'indice de l'extremum pour le contrôle
```

© 2000-2015, MetaQuotes Software Corp.

```
      ext=i-InpNum;
      //--- la vérification au maximum local
      if(IsMax(high,ext))
        {
         //--- marquons la bougie l'extremum
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=1;
         //--- marquons les autres bougies jusqu'à l'extremum par la couleur neutre
         FillCandles(open,high,low,close);
         //--- changeons les valeurs des variables
         ExtStart=ext+1;
         ExtCount=0;
         //--- passons à l'itération suivante
         continue;
        }
      //--- la vérification au minimum local
      if(IsMin(low,ext))
        {
         //--- marquons la bougie l'extremum
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=2;
         //--- marquons les autres bougies jusqu'à l'extremum par la couleur neutre
         FillCandles(open,high,low,close);
         //--- changeons les valeurs des variables
         ExtStart=ext+1;
         ExtCount=0;
         //--- passons à l'itération suivante
         continue;
        }
      //--- augmentons le nombre de bougies non extremums dans l'intervalle donné
      ExtCount++;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Si l'élément courant du tableau est le maximum locale            |
//+------------------------------------------------------------------+
bool IsMax(const double &price[],const int ind)
  {
//--- la variable du début de l'intervalle
   int i=ind-InpNum;
//--- la variable de la fin de l'intervalle
```

© 2000-2015, MetaQuotes Software Corp.

```
   int finish=ind+InpNum+1;
//--- la vérification pour la première moitié de l'intervalle
   for(;i<ind;i++)
     {
      if(price[ind]<=price[i])
         return(false);
     }
//--- la vérification pour la deuxième moitié de l'intervalle
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]<=price[i])
         return(false);
     }
//--- c'est l'extremum
   return(true);
  }
//+------------------------------------------------------------------+
//| Si l'élément courant du tableau est le minimum local             |
//+------------------------------------------------------------------+
bool IsMin(const double &price[],const int ind)
  {
//--- la variable du début de l'intervalle
   int i=ind-InpNum;
//--- la variable de la fin de l'intervalle
   int finish=ind+InpNum+1;
//--- la vérification pour la première moitié de l'intervalle
   for(;i<ind;i++)
     {
      if(price[ind]>=price[i])
         return(false);
     }
//--- la vérification pour la deuxième moitié de l'intervalle
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]>=price[i])
         return(false);
     }
//--- c'est l'extremum
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

ArrayCompare

Rend le résultat de la comparaison de deux tableux de même type. Peut être utilisé pour la comparaison des tableaux des types simples ou des structures d'utilisateur, n'ayant pas des objets complexes - c'est-à -dire qui ne contiennent pasdes chaînes, des tableaux dynamiques, des classes ou d'autres structures contenant des objets complexes.

```
int  ArrayCompare(
   const void&  array1[],           // le premier tableau
   const void&  array2[],           // le deuxième tableau
   int          start1=0,           // le décalage initial dans le premier tableau
   int          start2=0,           // le décalage initial dans le deuxième tableau
   int          count=WHOLE_ARRAY   // le nombre d'éléments pour la comparaison
   );
```

Réglages

array1[]

[in] Le premier tableau.

array2[]

[in] Le deuxième tableau.

start1=0

[in] L'indice initial de l'élément dans le premier tableau, par lequel commencera la comparaison. Par défaut l'indice de départ est - 0.

start2=0

[in] L'indice initial de l'élément dans le deuxième tableau, par lequel commencera la comparaison. Par défaut l'indice de départ est - 0.

count=WHOLE_ARRAY

[in] Le nombre d'éléments lesquels il faut comparer. Par défaut, tous les éléments des deux tableaux participent en comparaison (count=WHOLE_ARRAY).

La valeur rendue

- -1, si array1[] est moins que array2[]
- 0, si array1[] et array2[] sont égaux
- 1, si array1[] est plus grand que array2[]
- -2, à l'apparition de l'erreur à cause de l'incompatibilité des types des tableaux comparés, ou à la valeur start1, start2 ou count, amenant à la sortie au-delà du tableaux.

Note

Aux tailles différentes des tableaux comparés et à la valeur indiquée count=WHOLE_ARRAY pour le cas, quand un tableau est la sous-multitude exacte de l'autre, la fonction ne rendra pas 0 (les tableaux ne seront pas considérés égaux). Dans ce cas on rend le résultat de la comparaison des tailles de ces tableaux: -1, si la taille est array1[] moins aue la taille array2[] ou 1 dans le cas contraire.

© 2000-2015, MetaQuotes Software Corp.

ArrayFree

Libère la mémoire du tampon de n'importe quel tableau dynamique et met la dimension de la valeur zéro au 0.

```
void  ArrayFree(
   void&  array[]      // tableau
   );
```

Paramètres

array[]

[in] Le tableau dynamique.

La valeur rendue

Il n'y a pas de valeur rendue.

Note

Lors de l'écriture de scripts et d'indicateurs la nécessité de l'utilisation de la fonction ArrayFree() peut apparaître pas trop souvent: puisque à l'arrêt du travail d'un script toute la mémoire utilisée se libère tout de suite, et dans les indicateurs d'utilisateur le travail principal avec les tableaux représente l'accès aux tampons d'indicateur, dont les tailles sont dirigés automatiquement par le sous-système exécutant du terminal.

S'il est nécessaire de gérer la mémoire indépendamment dans les conditions complexes dynamiques, la fonction ArrayFree() permettra de désallouer manifestement et immédiatement la mémoire occupée par un tableau dynamique déjà inutile.

Exemple:

```
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
#include <Controls\Label.mqh>
#include <Controls\ComboBox.mqh>
//--- les constantes prédéterminées
#define X_START 0
#define Y_START 0
#define X_SIZE 280
#define Y_SIZE 300
//+------------------------------------------------------------------+
//| la classe du dialogue pour le travail avec la mémoire
//+------------------------------------------------------------------+
class CMemoryControl : public CAppDialog
  {
private:
   //--- la taille du tableau
   int               m_arr_size;
   //--- les tableaux
   char              m_arr_char[];
   int               m_arr_int[];
```

© 2000-2015, MetaQuotes Software Corp.

```
   float             m_arr_float[];
   double            m_arr_double[];
   long              m_arr_long[];
   //--- les inscriptions
   CLabel            m_lbl_memory_physical;
   CLabel            m_lbl_memory_total;
   CLabel            m_lbl_memory_available;
   CLabel            m_lbl_memory_used;
   CLabel            m_lbl_array_size;
   CLabel            m_lbl_array_type;
   CLabel            m_lbl_error;
   CLabel            m_lbl_change_type;
   CLabel            m_lbl_add_size;
   //--- les boutons
   CButton           m_button_add;
   CButton           m_button_free;
   //--- les listes
   CComboBox         m_combo_box_step;
   CComboBox         m_combo_box_type;
   //--- la valeur actuelle du type du tableau de la liste
   int               m_combo_box_type_value;

public:
                     CMemoryControl(void);
                    ~CMemoryControl(void);
   //--- la méthode de la création de l'objet de la classe
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- le gestionnaire des événements du graphique
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- la création de l'inscription
   bool              CreateLabel(CLabel &lbl,const string name,const int x,const int y
   //--- la création des éléments de la gestion
   bool              CreateButton(CButton &button,const string name,const int x,const
   bool              CreateComboBoxStep(void);
   bool              CreateComboBoxType(void);
   //--- les gestionnaires des événements
   void              OnClickButtonAdd(void);
   void              OnClickButtonFree(void);
   void              OnChangeComboBoxType(void);
   //--- Les méthodes du travail avec un tableau courant
   void              CurrentArrayFree(void);
   bool              CurrentArrayAdd(void);
  };
//+------------------------------------------------------------------+
//| La désallocation de la mémoire du tableau courant                |
//+------------------------------------------------------------------+
void CMemoryControl::CurrentArrayFree(void)
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//--- l' oblitération de la taille du tableau
   m_arr_size=0;
//--- la désallocation du tableau
   if(m_combo_box_type_value==0)
      ArrayFree(m_arr_char);
   if(m_combo_box_type_value==1)
      ArrayFree(m_arr_int);
   if(m_combo_box_type_value==2)
      ArrayFree(m_arr_float);
   if(m_combo_box_type_value==3)
      ArrayFree(m_arr_double);
   if(m_combo_box_type_value==4)
      ArrayFree(m_arr_long);
  }
//+------------------------------------------------------------------+
//| la tentative du supplément de la mémoire pour un tableau courant
//+------------------------------------------------------------------+
bool CMemoryControl::CurrentArrayAdd(void)
  {
//---si la taille de la mémoire utilisée est plus que la taille de la mémoire physique
   if(TerminalInfoInteger(TERMINAL_MEMORY_PHYSICAL)/TerminalInfoInteger(TERMINAL_MEMOR
      return(false);
//--- la tentative de l'allocation de la mémoire en fonction du type courant
   if(m_combo_box_type_value==0 && ArrayResize(m_arr_char,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==1 && ArrayResize(m_arr_int,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==2 && ArrayResize(m_arr_float,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==3 && ArrayResize(m_arr_double,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==4 && ArrayResize(m_arr_long,m_arr_size)==-1)
      return(false);
//--- la mémoire est allouée
   return(true);
  }
//+------------------------------------------------------------------+
//| Le traitement des événements                                     |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CMemoryControl)
ON_EVENT(ON_CLICK,m_button_add,OnClickButtonAdd)
ON_EVENT(ON_CLICK,m_button_free,OnClickButtonFree)
ON_EVENT(ON_CHANGE,m_combo_box_type,OnChangeComboBoxType)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Le constructeur                                                  |
//+------------------------------------------------------------------+
CMemoryControl::CMemoryControl(void)
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
  }
//+------------------------------------------------------------------+
//| Le destructeur                                                   |
//+------------------------------------------------------------------+
CMemoryControl::~CMemoryControl(void)
  {
  }
//+------------------------------------------------------------------+
//| La méthode de la création de l'objet de la classe
//+------------------------------------------------------------------+
bool CMemoryControl::Create(const long chart,const string name,const int subwin,
                            const int x1,const int y1,const int x2,const int y2)
  {
//--- la création de l'objet de la classe de base
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- la préparation des lignes pour les inscriptions
   string str_physical="Memory physical = "+(string)TerminalInfoInteger(TERMINAL_MEMOR
   string str_total="Memory total = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_TOTA
   string str_available="Memory available = "+(string)TerminalInfoInteger(TERMINAL_MEM
   string str_used="Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_USED)+
//--- la création des inscriptions
   if(!CreateLabel(m_lbl_memory_physical,"physical_label",X_START+10,Y_START+5,str_phy
      return(false);
   if(!CreateLabel(m_lbl_memory_total,"total_label",X_START+10,Y_START+30,str_total,12
      return(false);
   if(!CreateLabel(m_lbl_memory_available,"available_label",X_START+10,Y_START+55,str_
      return(false);
   if(!CreateLabel(m_lbl_memory_used,"used_label",X_START+10,Y_START+80,str_used,12,cl
      return(false);
   if(!CreateLabel(m_lbl_array_type,"type_label",X_START+10,Y_START+105,"Array type =
      return(false);
   if(!CreateLabel(m_lbl_array_size,"size_label",X_START+10,Y_START+130,"Array size =
      return(false);
   if(!CreateLabel(m_lbl_error,"error_label",X_START+10,Y_START+155,"",12,clrRed))
      return(false);
   if(!CreateLabel(m_lbl_change_type,"change_type_label",X_START+10,Y_START+185,"Chang
      return(false);
   if(!CreateLabel(m_lbl_add_size,"add_size_label",X_START+10,Y_START+210,"Add to arra
      return(false);
//--- la création des éléments de la gestion
   if(!CreateButton(m_button_add,"add_button",X_START+15,Y_START+245,"Add",12,clrBlue)
      return(false);
   if(!CreateButton(m_button_free,"free_button",X_START+75,Y_START+245,"Free",12,clrBl
      return(false);
   if(!CreateComboBoxType())
      return(false);
   if(!CreateComboBoxStep())
```

© 2000-2015, MetaQuotes Software Corp.

```
      return(false);
//--- l'initialisation de la variable
   m_arr_size=0;
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La création du bouton                                            |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateButton(CButton &button,const string name,const int x,
                                  const int y,const string str,const int font_size,
                                  const int clr)
  {
//--- la création du bouton
   if(!button.Create(m_chart_id,name,m_subwin,x,y,x+50,y+20))
      return(false);
//--- le texte
   if(!button.Text(str))
      return(false);
//--- la taille de la fonte
   if(!button.FontSize(font_size))
      return(false);
//--- la couleur de l'inscription
   if(!button.Color(clr))
      return(false);
//--- ajoutons le bouton dans les éléments du contrôle
   if(!Add(button))
      return(false);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La création de la liste pour la taille du tableau                |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxStep(void)
  {
//--- la création de la liste
   if(!m_combo_box_step.Create(m_chart_id,"step_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- le supplément des éléments à la liste
   if(!m_combo_box_step.ItemAdd("100 000",100000))
      return(false);
   if(!m_combo_box_step.ItemAdd("1 000 000",1000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("10 000 000",10000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("100 000 000",100000000))
      return(false);
//--- établissons l'élément courant de la liste
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!m_combo_box_step.SelectByValue(1000000))
      return(false);
//---ajoutons la liste aux éléments du contrôle
   if(!Add(m_combo_box_step))
      return(false);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La création de la liste pour le type du tableau                  |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxType(void)
  {
//--- la création de la liste
   if(!m_combo_box_type.Create(m_chart_id,"type_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- le supplément des éléments à la liste
   if(!m_combo_box_type.ItemAdd("char",0))
      return(false);
   if(!m_combo_box_type.ItemAdd("int",1))
      return(false);
   if(!m_combo_box_type.ItemAdd("float",2))
      return(false);
   if(!m_combo_box_type.ItemAdd("double",3))
      return(false);
   if(!m_combo_box_type.ItemAdd("long",4))
      return(false);
//--- établissons l'élément courant de la liste
   if(!m_combo_box_type.SelectByValue(3))
      return(false);
//--- rappelons l'élément courant de la liste
   m_combo_box_type_value=3;
//---ajoutons la liste aux éléments du contrôle
   if(!Add(m_combo_box_type))
      return(false);
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La création de l'inscription                                     |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateLabel(CLabel &lbl,const string name,const int x,
                                 const int y,const string str,const int font_size,
                                 const int clr)
  {
//--- la création de l'inscription
   if(!lbl.Create(m_chart_id,name,m_subwin,x,y,0,0))
      return(false);
//--- le texte
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!lbl.Text(str))
      return(false);
//--- la taille de la fonte
   if(!lbl.FontSize(font_size))
      return(false);
//--- la couleur
   if(!lbl.Color(clr))
      return(false);
//--- ajoutons l'inscription aux éléments du contrôle
   if(!Add(lbl))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Le gestionnaire de l'événement de la pression sur le bouton "Add"
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonAdd(void)
  {
//--- augmentons la taille du tableau
   m_arr_size+=(int)m_combo_box_step.Value();
//---essayons d'allouer la mémoire pour ce tableau courant
   if(CurrentArrayAdd())
     {
      //--- la mémoire est allouée, déduisons l'état courant sur l'écran
      m_lbl_memory_available.Text("Memory available = "+(string)TerminalInfoInteger(TE
      m_lbl_memory_used.Text("Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEM
      m_lbl_array_size.Text("Array size = "+IntegerToString(m_arr_size));
      m_lbl_error.Text("");
     }
   else
     {
      //--- on n'a pas réussi à allouer la mémoire, déduisons le message sur l'erreur
      m_lbl_error.Text("Array is too large, error!");
      //--- rendons la taille précédent du tableau
      m_arr_size-=(int)m_combo_box_step.Value();
     }
  }
//+------------------------------------------------------------------+
//| Le gestionnaire de l'événement de la pression sur le bouton "Free"
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonFree(void)
  {
//--- désallouons la mémoire du tableau courant
   CurrentArrayFree();
//--- déduisons l'état courant sur l'écran
   m_lbl_memory_available.Text("Memory available = "+(string)TerminalInfoInteger(TERMI
   m_lbl_memory_used.Text("Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEMORY
   m_lbl_array_size.Text("Array size = 0");
```

© 2000-2015, MetaQuotes Software Corp.

```
   m_lbl_error.Text("");
  }
//+------------------------------------------------------------------+
//| Le gestionnaire de l'événement du changement de la liste
//+------------------------------------------------------------------+
void CMemoryControl::OnChangeComboBoxType(void)
  {
//--- la vérification si le type du tableau a été modifié
   if(m_combo_box_type.Value()!=m_combo_box_type_value)
     {
      //--- désallouons la mémoire du tableau courant
      OnClickButtonFree();
      //--- travaillons avec un autre type du tableau
      m_combo_box_type_value=(int)m_combo_box_type.Value();
      //--- déduisons un nouveau type du tableau sur l'écran
      if(m_combo_box_type_value==0)
         m_lbl_array_type.Text("Array type = char");
      if(m_combo_box_type_value==1)
         m_lbl_array_type.Text("Array type = int");
      if(m_combo_box_type_value==2)
         m_lbl_array_type.Text("Array type = float");
      if(m_combo_box_type_value==3)
         m_lbl_array_type.Text("Array type = double");
      if(m_combo_box_type_value==4)
         m_lbl_array_type.Text("Array type = long");
     }
  }
//--- l'objet de la classe CMemoryControl
CMemoryControl ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- la création du dialogue
   if(!ExtDialog.Create(0,"MemoryControl",0,X_START,Y_START,X_SIZE,Y_SIZE))
      return(INIT_FAILED);
//--- le lancement
   ExtDialog.Run();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ExtDialog.Destroy(reason);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2015, MetaQuotes Software Corp.

ArrayGetAsSeries

Contrôle la direction de l'indexation du tableau .

```
bool  ArrayGetAsSeries(
   const void&  array[]    // tableau pour le contrôle
   );
```

Paramètres

array

[in] Le tableau contrôlé.

La valeur rendue

Rend true, si le drapeau AS_SERIES est établit au tableau indiqué,c'est-à-dire l'accès au tableau est exécuté de sa fin au commencement comme dans la série temporelle. La série temporelle se distingue du tableau ordinaire par ce que l'indexation des éléments de la série temporelle est produite de la fin du tableau vers le début (des plus nouvelles données aux plus vieilles).

Note

Pour le contrôle du tableau de l'appartenance à la série temporelle il faut appliquer la fonction ArrayIsSeries(). Les tableaux des données de prix transmises à titre des paramètres d'entrée à la fonction OnCalculate(),ont pas absolument la direction de l'indexation comme aux séries temporelles. On peut établir la direction nécessaire de l'indexation par la fonction ArraySetAsSeries().

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
#property description "L'indicateur calcule les valeurs absolues de la différence entr
#property description "Open et Close ou High et Low, et les affiche dans une fenêtre s
#property description "comme un histogramme."
//--- les réglages de l'indicateur
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- les paramètres d'entrée
input bool InpAsSeries=true; // La direction de l'indexation dans le tampon d'indicate
input bool InpPrices=true;   // Les prix pour le calcul (true - Open,Close; false - Hi
//--- le tampon d'indicateur
double ExtBuffer[];
//+------------------------------------------------------------------+
//| Le calcul des valeurs de l'indicateur                            |
//+------------------------------------------------------------------+
void CandleSizeOnBuffer(const int rates_total,const int prev_calculated,
                        const double &first[],const double &second[],double &buffer[])
  {
//--- la variable du début pour le calcul des barres
   int start=prev_calculated;
//---si les valeurs de l'indicateur étaient déjà calculées sur un tick précédent, trav
   if(prev_calculated>0)
      start--;
//--- définissons la direction de l'indexation dans les tableaux
   bool as_series_first=ArrayGetAsSeries(first);
   bool as_series_second=ArrayGetAsSeries(second);
   bool as_series_buffer=ArrayGetAsSeries(buffer);
//--- changeons la direction de l'indexation sur direct, si c'est nécessaire
   if(as_series_first)
      ArraySetAsSeries(first,false);
   if(as_series_second)
      ArraySetAsSeries(second,false);
   if(as_series_buffer)
      ArraySetAsSeries(buffer,false);
//--- calculons les valeurs de l'indicateur
   for(int i=start;i<rates_total;i++)
      buffer[i]=MathAbs(first[i]-second[i]);
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tampons d'indicateur
   SetIndexBuffer(0,ExtBuffer);
//--- établissons la direction de l'indexation dans le tampon d'indicateur
   ArraySetAsSeries(ExtBuffer,InpAsSeries);
//--- vérifions pour quels prix l'indicateur est calculé
   if(InpPrices)
     {
      //--- les prix Open et Close
      PlotIndexSetString(0,PLOT_LABEL,"BodySize");
      //--- établissons la couleur de l'indicateur
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrOrange);
     }
   else
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- les prix High et Low
      PlotIndexSetString(0,PLOT_LABEL,"ShadowSize");
      //--- établissons la couleur de l'indicateur
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrDodgerBlue);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le calcul de l'indicateur en fonction de la valeur du drapeau
   if(InpPrices)
      CandleSizeOnBuffer(rates_total,prev_calculated,open,close,ExtBuffer);
   else
      CandleSizeOnBuffer(rates_total,prev_calculated,high,low,ExtBuffer);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

L'accès aux séries temporelles, ArraySetAsSeries

© 2000-2015, MetaQuotes Software Corp.

ArrayInitialize

Initialise le tableau numérique par la signification indiquée.

For initialization of an array of char type

```
int  ArrayInitialize(
   char    array[],     // tableau initialisé
   char    value        // la valeur qui sera établie
   );
```

For initialization of an array of short type

```
int  ArrayInitialize(
   short   array[],     // tableau initialisé
   short   value        // la valeur qui sera établie
   );
```

For initialization of an array of int type

```
int  ArrayInitialize(
   int     array[],     // tableau initialisé
   int     value        // la valeur qui sera établie
   );
```

For initialization of an array of long type

```
int  ArrayInitialize(
   long    array[],     // tableau initialisé
   long    value        // la valeur qui sera établie
   );
```

For initialization of an array of float type

```
int  ArrayInitialize(
   float   array[],     // tableau initialisé
   float   value        // la valeur qui sera établie
   );
```

For initialization of an array of double type

```
int  ArrayInitialize(
   double  array[],     // tableau initialisé
   double  value        // la valeur qui sera établie
   );
```

For initialization of an array of bool type

```
int  ArrayInitialize(
   bool    array[],     // tableau initialisé
   bool    value        // la valeur qui sera établie
   );
```

© 2000-2015, MetaQuotes Software Corp.

For initialization of an array of uint type

```
int  ArrayInitialize(
   uint    array[],     // tableau initialisé
   uint    value        // la valeur qui sera établie
   );
```

Paramètres

array[]

[out] Le tableau numérique, qu'il faut initialiser.

value

[in] Une nouvelle valeur, qu'il faut établir à tous les éléments du tableau.

La valeur rendue

Le nombre d'éléments.

Note

La fonction ArrayResize() permet de mettre la grandeur du tableau avec une certaine réserve pour une future augmentation sans la réallocation physique de la mémoire. C'est fait pour l'amélioration de la rapidité puisque les opérations de la distribution de la mémoire sont assez lentes.

L'initialisation du tableau par l'expression ArrayInitialize(array, init_val) ne signifie pas l'initialisation par la même valeur et les éléments de la réserve, allouée pour ce tableau. Aux augmentations ultérieures du grandeur du tableau *array* par la fonction ArrayResize() dans la limite de la réserve courante, à la fin du tableau sont ajoutés les éléments, dont les valeur ne sont pas définies et, le plus souvent, ne sont pas égaux à *init_val*.

Exemple:

```
void OnStart()
  {
//--- tableau dynamique
   double array[];
//--- spécifions la grandeur du tableau pour 100 éléments et réservons encore un tampo
   ArrayResize(array,100,10);
//--- initialisons les éléments du tableau par la valeur EMPTY_VALUE=DBL_MAX
   ArrayInitialize(array,EMPTY_VALUE);
   Print("Valeurs des derniers 10 éléments du tableau après l'initialisation");
   for(int i=90;i<100;i++) printf("array[%d]=%G",i,array[i]);
//--- augmentons la grandeur du tableau pour 5 éléments
   ArrayResize(array,105);
   Print("Valeur des derniers 10 éléments du tableau après ArrayResize(array,105)");
//--- valeurs des derniers 5 éléments étaient reçues du tampon de la réserve
   for(int i=95;i<105;i++) printf("array[%d] = %G",i,array[i]);
  }
```

© 2000-2015, MetaQuotes Software Corp.

ArrayFill

Remplit un tableau numérique par la valeur indiquée.

```
void  ArrayFill(
   void&  array[],       // le tableau
   int    start,         // l'index de l'élément initial
   int    count,         // le nombre d'éléments
   void   value          // la valeur, par laquelle on remplit le tableau
   );
```

Paramètres

array[]

[out] Un tableau du type simple (char, uchar, short, ushort, int, uint, long, ulong, bool, color, datetime, float, double).

start

[in] L'index de l'élément initial (par quel élément il faut remplir). N'est pas pris en considérationle drapeau de série établi.

count

[in] Le nombre d'éléments qui doivent être remplis.

value

[in] La valeur, par laquelle on remplit le tableau.

La valeur rendue

Il n'y a pas de valeur rendue.

Note

A l'appel de la fonction ArrayFill() on sous-entend toujours la direction ordinaire de l'indexation - de gauche à droite, c'est-à-dire le changement de l'ordre d'accès vers les éléments du tableau à l'aide de la fonction ArraySetAsSeries() n'est pas pris en considération.

Le tableau multidimensionnel au traitement par la fonction ArrayFill() se présente comme unidimentionnel, par exemple, le tableau array[2][4] est traité comme array[8], c'est pourquoi au travail avec ce tableau est admissible d'indiquer l'indice de l'élément initial égal au 5. Ainsi l'appel ArrayFill(array, 5, 2, 3.14) pour le tableau array[2][4] remplira par la valeur 3.14 les éléments du tableau array[1][1] et array[1][2].

Exemple:

```
void OnStart()
  {
//--- annonçons le tableau dynamique
   int a[];
//---définissons la taille
   ArrayResize(a,10);
//--- remplissons les 5 éléments initiaux par la valeurs 123
   ArrayFill(a,0,5,123);
//--- remplissons 5 éléments (à partir de 5-ème) par la valeur 456
   ArrayFill(a,5,5,456);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- déduisons les valeurs de tous les éléments
   for(int i=0;i<ArraySize(a);i++) printf("a[%d] = %d",i,a[i]);
  }
```

© 2000-2015, MetaQuotes Software Corp.

ArrayIsDynamic

Vérifie - si le tableau est dynamique.

```
bool  ArrayIsDynamic(
   const void&  array[]    // tableau contrôlé
   );
```

Paramètres

array[]

[in] Le tableau contrôlé.

La valeur rendue

Rend true, si le tableau est dynamique, autrement revient false.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
#property description ""Cet indicateur ne calcule pas les valeurs, il tente d'applique
#property description "l'appel de la fonction ArrayFree() à trois tableaux: au tampon
#property description "d'indicateur. Les résultats sont déduits au journal des experts
//--- les réglages de l'indicateur
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- les variables globales
double ExtDynamic[];   // le tableau dynamique
double ExtStatic[100]; // un tableau statique
bool   ExtFlag=true;   // le drapeau
double ExtBuff[];      // le tampon d'indicateur
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- allouons la mémoire pour le tableau
   ArrayResize(ExtDynamic,100);
//--- indicator buffers mapping
   SetIndexBuffer(0,ExtBuff);
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])
  {
//--- faisons l'analyse une fois
   if(ExtFlag)
     {
      //--- essayons de désallouer la mémoire pour les tableaux
      //--- 1. Un tableau dynamique
      Print("+============================+");
      Print("1. La vérification du tableau dynamique:");
      Print("La taille avant la désallocation de la mémoire = ",ArraySize(ExtDynamic))
      Print("C'est un tableau dynamique = ",ArrayIsDynamic(ExtDynamic) ? "Oui" : "Non"
      //--- tentons de désallouer la mémoire du tableau
      ArrayFree(ExtDynamic);
      Print("La taille après la désallocation de la mémoire = ",ArraySize(ExtDynamic))
      //--- 2. Un tableau statique
      Print("2. La vérification du tableau statique:");
      Print("La taille avant la désallocation de la mémoire = ",ArraySize(ExtStatic));
      Print("C'est un tableau statique = ",ArrayIsDynamic(ExtStatic) ? "Oui" : "Non");
      //--- tentons de désallouer la mémoire du tableau
      ArrayFree(ExtStatic);
      Print("La taille après la désallocation de la mémoire = ",ArraySize(ExtStatic));
      //--- 3. Le tampon d'indicateur
      Print("3. La vérification du tampon d'indicateur:");
      Print("La taille avant la désallocation de la mémoire = ",ArraySize(ExtBuff));
      Print("C'est un tableau dynamique = ",ArrayIsDynamic(ExtBuff) ? "Oui" : "Non");
      //--- tentons de désallouer la mémoire du tableau
      ArrayFree(ExtBuff);
      Print("La taille après la désallocation de la mémoire = ",ArraySize(ExtBuff));
      //--- changeons la valeur du drapeau
      ExtFlag=false;
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

L'accès aux séries temporelles et aux indicateurs

© 2000-2015, MetaQuotes Software Corp.

ArrayIsSeries

Vérifie - si le tableau est la série temporelle.

```
bool  ArrayIsSeries(
   const void&  array[]    // tableau contrôlé
   );
```

Paramètres

array[]

[in] Le tableau contrôlé.

La valeur rendue

True revient, si le tableau contrôlé est le tableau-la série temporelle, autrement se revient false. Il est nécessaire de contrôler les tableaux transmis à titre du paramètre de la fonction OnCalculate(),à l'accès aux éléments du tableau par la fonction ArrayGetAsSeries().

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   if(ArrayIsSeries(open))
      Print("open[] is timeseries");
   else
      Print("open[] is not timeseries!!!");
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

L'accès aux séries temporelles et aux indicateurs

© 2000-2015, MetaQuotes Software Corp.

ArrayMaximum

Ищет максимальный элемент в первом измерении многомерного числового массива.

```
int  ArrayMaximum(
   const void&   array[],            // tableau pour la recherche
   int           start=0,            // par quel index on commence la recherche
   int           count=WHOLE_ARRAY   // nombre de contrôlés
   );
```

Paramètres

array[]

[in] Le tableau numérique, dans laquelle la recherche est faite.

start=0

[in] L'index initial pour la recherche.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour la recherche. Par défaut, cherche dans tout le tableau (count=WHOLE_ARRAY).

La valeur rendue

La fonction rend un index d'un élément trouvé tenant compte du tableau sérial. En cas de l'échec la fonction rend -1.

Note

Поиск максимального элемента производится с учетом значения флага AS_SERIES.

Функции ArrayMaximum и ArrayMinimum принимают в качестве параметра массив любой размерности, при этом поиск происходит только по первому (нулевому) измерению.

Exemple:

```
#property description "L'indicateur affiche les bougies du temps trame supérieur sur c
//--- les réglages de l'indicateur
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- plot 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- plot 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
```

© 2000-2015, MetaQuotes Software Corp.

```
//---- plot 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- plot 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- plot 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- une constante prédéfinie
#define INDICATOR_EMPTY_VALUE 0.0
//--- les paramètres d'entrée
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Le temps trame pour le calc
input datetime        InpDateStart=D'2013.01.01 00:00'; // La date de commencement de
//--- les tampons d'indicateur pour les bougies d'ours
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- les tampons d'indicateur pour les bougies de taureau
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- les variables globales
datetime ExtTimeBuff[];      // le tampon pour le temps du temps trame supérieur
int      ExtSize=0;          // la taille du tampon du temps
int      ExtCount=0;         // l'indice à l'intérieur du tampon du temps
int      ExtStartPos=0;      // la position initiale pour le calcul de l'indicateur
bool     ExtStartFlag=true;  // le drapeau auxiliaire pour la réception de la positior
datetime ExtCurrentTime[1];  // le dernier temps de la formation de la barre du temps
datetime ExtLastTime;        // le dernier temps du temps trame supérieur, pour qui or
bool     ExtBearFlag=true;   // le drapeau pour la définition de l'ordre de l'enregist
bool     ExtBullFlag=true;   // le drapeau pour la définition de l'ordre de l'enregist
int      ExtIndexMax=0;      // l'indice de l'élément maximum dans un tableau
int      ExtIndexMin=0;      // l'indice de l'élément minimal dans un tableau
int      ExtDirectionFlag=0; // la direction du mouvement du prix à la bougie actuelle
//--- l'alinéa entre le prix de l'ouverture et de la clôture de la bougie pour la renc
```

© 2000-2015, MetaQuotes Software Corp.

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
//| Le coloriage de la partie principale de la bougie                |
//+------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- trouvons les indices des éléments maximums et minimaux dans les tableaux
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // le maximum dans High
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // le minimum dans Low
//--- déterminons le nombre de barres de la période actuelle on va colorier
   int count=fill_index-start+1;
//--- si le prix de la clôture sur la première barre est plus que le pris de la clôtur
   if(open[start]>close[last])
     {
      //--- si avant la bougie était haussière, effaçons les valeurs de tampons d'indi
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- la bougie baissière
      ExtDirectionFlag=-1;
      //--- formons la bougie
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- la sortie de la fonction
      return;
     }
//--- si le prix de la clôture de la première barre est plus petit que le prix de la c
   if(open[start]<close[last])
     {
      //---si avant la bougie était baissière, effaçons les valeurs de tampons d'indic
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- la bougie haussière
      ExtDirectionFlag=1;
      //--- formons la bougie
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- la sortie de la fonction
      return;
     }
//---si vous etes dans cette partie de la fonction, donc le prix de l'ouverture sur la
//--- au prix de la clôture sur la dernière barre, supposons que cette bougie est bais
//--- si avant la bougie était haussière, effaçons les valeurs de tampons d'indicateur
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- la bougie baissière
   ExtDirectionFlag=-1;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- si les prix de la clôture et le prix de l'ouverture sont égaux, utilisons le déc
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShac
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShac
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//| colorons la fin de la bougie                                     |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- si la barre est seule, alors ne dessinons pas
   if(last-start==0)
      return;
//--- si le prix de la clôture sur la première barre est plus que le pris de la clôtur
   if(open[start]>close[last])
     {
      //--- formons la fin de la bougie
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- la sortie de la fonction
      return;
     }
//--- si le prix de la clôture de la première barre est plus petit que le prix de la c
   if(open[start]<close[last])
     {
      //--- formons la fin de la bougie
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- la sortie de la fonction
      return;
     }
//---si vous etes dans cette partie de la fonction, donc le prix de l'ouverture sur la
//--- au prix de la clôture sur la dernière barre, supposons que cette bougie est bais
//--- formons la fin de la bougie
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_i
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEm
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- la vérification de la période de l'indicateur
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- l'affichage des données de prix au premier plan
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- le rattachement des tampons d'indicateur
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- spécifions certaines valeurs des propriétés pour la construction de l'indicateur
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // le type de la constructio
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // le style de la ligne de l
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // l'épaisseur de la ligne d
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//---s'il n'y a pas encore de barres calculés
   if(prev_calculated==0)
     {
      //--- recevons le temps de l'apparition des barres du temps trame supérieur
      if(!GetTimeData())
         return(0);
     }
//--- établissons la direction directe de l'indexation
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- la variable du début pour calculer les barres
   int start=prev_calculated;
//--- si la barre est formée, recalculons la valeur de l'indicateur sur lui
   if(start!=0 && start==rates_total)
      start--;
//--- le cycle du calcule des valeurs de l'indicateur
   for(int i=start;i<rates_total;i++)
     {
      //--- remplissons les I-èmes éléments des tampons d'indicateur par les valeurs v
      FillIndicatorBuffers(i);
      //--- passons le calcul pour les barres à partir de la date InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- définissons pour la première fois la position par laquelle on commence
         if(ExtStartFlag)
           {
            //--- rappelons le numéro de la barre initiale
            ExtStartPos=i;
            //---définissons la première date du temps trame supérieur qui est plus qu
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- changeons la valeurs du drapeau pour ne se trouver plus dans ce bloc
            ExtStartFlag=false;
           }
         //--- la vérification s'il y a encore des éléments dans le tableau
         if(ExtCount<ExtSize)
           {
            //--- attendons, quand la valeur du temps du temps trame courant atteindra
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- dessinons la partie principale de la bougie (sans coloriage entre
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,ExtI
               //--- colorions la fin de la bougie (le domaine entre la dernière et av
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
```

© 2000-2015, MetaQuotes Software Corp.

```
                //--- rapprochons la position initiale pour le dessin de la bougie suiv
                ExtStartPos=i;
                //--- augmentons le compteur du tableau
                ExtCount++;
               }
             else
                continue;
            }
          else
            {
             //--- oblitérons la valeur de l'erreur
             ResetLastError();
             //--- recevons la dernière date du temps trame supérieur
             if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("L'erreur du copiage des données, le code = ",GetLastError());
                return(0);
               }
             //--- si une nouvelle date est plus grande, donc finissons la formation de
             if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- effaçons le domaine entre la dernière et avant-dernière barre dan
                ClearEndOfBodyMain(i-1);
                //---colorions ce domaine à l'aide des tampons auxiliaires d'indicateur
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- rapprochons la position initiale pour le dessin de la bougie suiv
                ExtStartPos=i;
                //--- oblitérons le drapeau de la direction du prix
                ExtDirectionFlag=0;
                //---retenons une nouvelle dernière date
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- formons la bougie
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| La vérification de la période introduite de l'indicateur de la convenance
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- la période de l'indicateur doit être plus que le temps trame, sur lequel il s'ai
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(current_period>=high_period)
     {
      Print("L'erreur! La valeur de la période de l'indicateur doit être plus que de v
      return(false);
     }
//--- si la période de l'indicateur - une semaine ou un mois, la période est correcte
   if(high_period>32768)
      return(true);
//--- amenons les valeurs des périodes aux minutes
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- la période de l'indicateur doit être multiple au temps trame, sur lequel il s'af
   if(high_period%current_period!=0)
     {
      Print("L'erreur! La valeur de la période de l'indicateur doit être multiple à la
      return(false);
     }
//--- la période de l'indicateur doit excéder la valeur du temps trame, sur lequel il
   if(high_period/current_period<3)
     {
      Print("L'erreur! La valeur de la période de l'indicateur doit excéder la valeur
      return(false);
     }
//--- la période de l'indicateur est correcte pour le temps trame courant
   return(true);
  }
//+------------------------------------------------------------------+
//| La réception des données du temps du temps trame supérieur       |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- l'oblitération de la valeur de l'erreur
   ResetLastError();
//--- copions toutes les données pour un temps courant
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- recevons le code de l'erreur
      int code=GetLastError();
      //--- imprimons le texte de l'erreur
      PrintFormat("L'erreur du copiage des données! %s",code==4401
                  ? "L'histoire est encore chargée!"
                  : "Le code = "+IntegerToString(code));
      //--- rendons false pour la tentative réitérée du chargement des données
      return(false);
     }
//--- recevons la taille du tableau
   ExtSize=ArraySize(ExtTimeBuff);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- établissons l'indice du cycle pour le tableau égal au zéro
   ExtCount=0;
//--- établissons la position de la bougie courante qui est égal au zéro sur le temps
   ExtStartPos=0;
   ExtStartFlag=true;
//--- retenons la dernière valeur du temps du temps trame supérieur
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction forme la partie principale de la bougie. En fonction de la valeur|
//| du drapeau, la fonction définit, quelles données doivent  |
//| être enregistrées dans quels tableaux pour l'affichage correct .
//+------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- vérifions la valeur de l'indicateur
   if(flag)
     {
      //--- formons le corps de la bougie
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- formons l'ombre de la bougie
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- formons le corps de la bougie
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- formons l'ombre de la bougie
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------+
//| La fonction forme la fin de la bougie. En fonction de la valeur du drapeau,  |
//| la fonction définit, quelles données doivent           |
//| être enregistrées dans quels tableaux pour l'affichage correct .
//+------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- vérifions la valeur de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(flag)
     {
      //--- formons la fin du corps de la bougie
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- formons la fin de l'ombre de la bougie
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- changeons la valeur du drapeau à l'opposé
      flag=false;
     }
   else
     {
      //--- formons la fin du corps de la bougie
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- formons la fin de l'ombre de la bougie
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- changeons la valeur du drapeau à l'opposé
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Le nettoyage de la fin de la bougie (le domaine entre la dernière et avant-dernièr
//| barre)                                                           |
//+------------------------------------------------------------------+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------+
//| Le nettoyage de la bougie                                        |
//+------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- la vérification
   if(count!=0)
     {
      //--- remplissons les tampons d'indicateur par la valeur vide
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------+
//| La formation de la partie principale de la bougie                |
//+------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//--- la vérification
   if(count!=0)
     {
      //--- remplissons les tampons d'indicateur par les valeurs
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------+
//| La formation de la fin de la bougie                              |
//+------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- remplissons les tampons d'indicateur par les valeurs
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Remplissons du I-ème élément des tampons d'indicateur par les valeurs vides|
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- établissons la valeur vide à la cellule des tampons d'indicateur
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000-2015, MetaQuotes Software Corp.

ArrayMinimum

Ищет минимальный элемент в первом измерении многомерного числового массива.

```
int  ArrayMinimum(
   const void&   array[],            // tableau pour la recherche
   int           start=0,            // par quel index on commence la recherche
   int           count=WHOLE_ARRAY   // nombre de contrôlés
   );
```

Paramètres

array[]

[in] Le tableau numérique, dans laquelle la recherche est faite.

start=0

[in] L'index initial pour la recherche.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour la recherche. Par défaut, cherche dans tout le tableau (count=WHOLE_ARRAY).

La valeur rendue

La fonction rend un index d'un élément trouvé tenant compte du tableau sérial. En cas de l'échec la fonction rend -1.

Note

Поиск минимального элемента производится с учетом значения флага AS_SERIES.

Функции ArrayMaximum и ArrayMinimum принимают в качестве параметра массив любой размерности, при этом поиск происходит только по первому (нулевому) измерению.

Exemple:

```
#property description "L'indicateur affiche les bougies du temps trame supérieur sur c
//--- les réglages de l'indicateur
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- plot 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- plot 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
```

© 2000-2015, MetaQuotes Software Corp.

```
//---- plot 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- plot 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- plot 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- une constante prédéfinie
#define INDICATOR_EMPTY_VALUE 0.0
//--- les paramètres d'entrée
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Le temps trame pour le calc
input datetime        InpDateStart=D'2013.01.01 00:00'; // La date de commencement de
//--- les tampons d'indicateur pour les bougies d'ours
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- les tampons d'indicateur pour les bougies de taureau
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- les variables globales
datetime ExtTimeBuff[];      // le tampon pour le temps du temps trame supérieur
int      ExtSize=0;          // la taille du tampon du temps
int      ExtCount=0;         // l'indice à l'intérieur du tampon du temps
int      ExtStartPos=0;      // la position initiale pour le calcul de l'indicateur
bool     ExtStartFlag=true;  // le drapeau auxiliaire pour la réception de la positior
datetime ExtCurrentTime[1];  // le dernier temps de la formation de la barre du temps
datetime ExtLastTime;        // le dernier temps du temps trame supérieur, pour qui or
bool     ExtBearFlag=true;   // le drapeau pour la définition de l'ordre de l'enregist
bool     ExtBullFlag=true;   // le drapeau pour la définition de l'ordre de l'enregist
int      ExtIndexMax=0;      // l'indice de l'élément maximum dans un tableau
int      ExtIndexMin=0;      // l'indice de l'élément minimal dans un tableau
int      ExtDirectionFlag=0; // la direction du mouvement du prix à la bougie actuelle
//--- l'alinéa entre le prix de l'ouverture et de la clôture de la bougie pour la renc
```

© 2000-2015, MetaQuotes Software Corp.

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
//| Le coloriage de la partie principale de la bougie                |
//+------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- trouvons les indices des éléments maximums et minimaux dans les tableaux
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // le maximum dans High
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // le minimum dans Low
//--- déterminons le nombre de barres de la période actuelle on va colorier
   int count=fill_index-start+1;
//--- si le prix de la clôture sur la première barre est plus que le pris de la clôtu
   if(open[start]>close[last])
     {
      //--- si avant la bougie était haussière, effaçons les valeurs de tampons d'indi
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- la bougie baissière
      ExtDirectionFlag=-1;
      //--- formons la bougie
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- la sortie de la fonction
      return;
     }
//--- si le prix de la clôture de la première barre est plus petit que le prix de la c
   if(open[start]<close[last])
     {
      //---si avant la bougie était baissière, effaçons les valeurs de tampons d'indic
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- la bougie haussière
      ExtDirectionFlag=1;
      //--- formons la bougie
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- la sortie de la fonction
      return;
     }
//---si vous etes dans cette partie de la fonction, donc le prix de l'ouverture sur la
//--- au prix de la clôture sur la dernière barre, supposons que cette bougie est bais
//--- si avant la bougie était haussière, effaçons les valeurs de tampons d'indicateur
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- la bougie baissière
   ExtDirectionFlag=-1;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- si les prix de la clôture et le prix de l'ouverture sont égaux, utilisons le déc
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShac
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShac
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//| colorons la fin de la bougie                                     |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- si la barre est seule, alors ne dessinons pas
   if(last-start==0)
      return;
//--- si le prix de la clôture sur la première barre est plus que le pris de la clôtur
   if(open[start]>close[last])
     {
      //--- formons la fin de la bougie
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- la sortie de la fonction
      return;
     }
//--- si le prix de la clôture de la première barre est plus petit que le prix de la c
   if(open[start]<close[last])
     {
      //--- formons la fin de la bougie
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- la sortie de la fonction
      return;
     }
//---si vous etes dans cette partie de la fonction, donc le prix de l'ouverture sur la
//--- au prix de la clôture sur la dernière barre, supposons que cette bougie est bais
//--- formons la fin de la bougie
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_i
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEm
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- la vérification de la période de l'indicateur
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- l'affichage des données de prix au premier plan
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- le rattachement des tampons d'indicateur
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- spécifions certaines valeurs des propriétés pour la construction de l'indicateur
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // le type de la constructio
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // le style de la ligne de l
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // l'épaisseur de la ligne d
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//---s'il n'y a pas encore de barres calculés
   if(prev_calculated==0)
     {
      //--- recevons le temps de l'apparition des barres du temps trame supérieur
      if(!GetTimeData())
         return(0);
     }
//--- établissons la direction directe de l'indexation
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- la variable du début pour calculer les barres
   int start=prev_calculated;
//--- si la barre est formée, recalculons la valeur de l'indicateur sur lui
   if(start!=0 && start==rates_total)
      start--;
//--- le cycle du calcule des valeurs de l'indicateur
   for(int i=start;i<rates_total;i++)
     {
      //--- remplissons les I-èmes éléments des tampons d'indicateur par les valeurs v
      FillIndicatorBuffers(i);
      //--- passons le calcul pour les barres à partir de la date InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- définissons pour la première fois la position par laquelle on commence
         if(ExtStartFlag)
           {
            //--- rappelons le numéro de la barre initiale
            ExtStartPos=i;
            //---définissons la première date du temps trame supérieur qui est plus qu
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- changeons la valeurs du drapeau pour ne se trouver plus dans ce bloc
            ExtStartFlag=false;
           }
         //--- la vérification s'il y a encore des éléments dans le tableau
         if(ExtCount<ExtSize)
           {
            //--- attendons, quand la valeur du temps du temps trame courant atteindra
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- dessinons la partie principale de la bougie (sans coloriage entre
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,ExtI
               //--- colorions la fin de la bougie (le domaine entre la dernière et av
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
```

© 2000-2015, MetaQuotes Software Corp.

```
                //--- rapprochons la position initiale pour le dessin de la bougie suiv
                ExtStartPos=i;
                //--- augmentons le compteur du tableau
                ExtCount++;
               }
             else
                continue;
            }
          else
            {
             //--- oblitérons la valeur de l'erreur
             ResetLastError();
             //--- recevons la dernière date du temps trame supérieur
             if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("L'erreur du copiage des données, le code = ",GetLastError());
                return(0);
               }
             //--- si une nouvelle date est plus grande, donc finissons la formation de
             if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- effaçons le domaine entre la dernière et avant-dernière barre dan
                ClearEndOfBodyMain(i-1);
                //---colorions ce domaine à l'aide des tampons auxiliaires d'indicateur
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- rapprochons la position initiale pour le dessin de la bougie suiv
                ExtStartPos=i;
                //--- oblitérons le drapeau de la direction du prix
                ExtDirectionFlag=0;
                //---retenons une nouvelle dernière date
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- formons la bougie
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| La vérification de la période introduite de l'indicateur de la convenance
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- la période de l'indicateur doit être plus que le temps trame, sur lequel il s'ai
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(current_period>=high_period)
     {
      Print("L'erreur! La valeur de la période de l'indicateur doit être plus que de v
      return(false);
     }
//--- si la période de l'indicateur - une semaine ou un mois, la période est correcte
   if(high_period>32768)
      return(true);
//--- amenons les valeurs des périodes aux minutes
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- la période de l'indicateur doit être multiple au temps trame, sur lequel il s'af
   if(high_period%current_period!=0)
     {
      Print("L'erreur! La valeur de la période de l'indicateur doit être multiple à la
      return(false);
     }
//--- la période de l'indicateur doit excéder la valeur du temps trame, sur lequel il
   if(high_period/current_period<3)
     {
      Print("L'erreur! La valeur de la période de l'indicateur doit excéder la valeur
      return(false);
     }
//--- la période de l'indicateur est correcte pour le temps trame courant
   return(true);
  }
//+------------------------------------------------------------------+
//| La réception des données du temps du temps trame supérieur       |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- l'oblitération de la valeur de l'erreur
   ResetLastError();
//--- copions toutes les données pour un temps courant
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- recevons le code de l'erreur
      int code=GetLastError();
      //--- imprimons le texte de l'erreur
      PrintFormat("L'erreur du copiage des données! %s",code==4401
                  ? "L'histoire est encore chargée!"
                  : "Le code = "+IntegerToString(code));
      //--- rendons false pour la tentative réitérée du chargement des données
      return(false);
     }
//--- recevons la taille du tableau
   ExtSize=ArraySize(ExtTimeBuff);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- établissons l'indice du cycle pour le tableau égal au zéro
   ExtCount=0;
//--- établissons la position de la bougie courante qui est égal au zéro sur le temps
   ExtStartPos=0;
   ExtStartFlag=true;
//--- retenons la dernière valeur du temps du temps trame supérieur
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- l'exécution réussie
   return(true);
  }
//+------------------------------------------------------------------+
//| La fonction forme la partie principale de la bougie. En fonction de la valeur|
//| du drapeau, la fonction définit, quelles données doivent  |
//| être enregistrées dans quels tableaux pour l'affichage correct .
//+------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- vérifions la valeur de l'indicateur
   if(flag)
     {
      //--- formons le corps de la bougie
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- formons l'ombre de la bougie
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- formons le corps de la bougie
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- formons l'ombre de la bougie
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------+
//| La fonction forme la fin de la bougie. En fonction de la valeur du drapeau,  |
//| la fonction définit, quelles données doivent           |
//| être enregistrées dans quels tableaux pour l'affichage correct .
//+------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- vérifions la valeur de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(flag)
     {
      //--- formons la fin du corps de la bougie
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- formons la fin de l'ombre de la bougie
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- changeons la valeur du drapeau à l'opposé
      flag=false;
     }
   else
     {
      //--- formons la fin du corps de la bougie
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- formons la fin de l'ombre de la bougie
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- changeons la valeur du drapeau à l'opposé
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Le nettoyage de la fin de la bougie (le domaine entre la dernière et avant-dernièr
//| barre)                                                           |
//+------------------------------------------------------------------+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------+
//| Le nettoyage de la bougie                                        |
//+------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- la vérification
   if(count!=0)
     {
      //--- remplissons les tampons d'indicateur par la valeur vide
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------+
//| La formation de la partie principale de la bougie                |
//+------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//--- la vérification
   if(count!=0)
     {
      //--- remplissons les tampons d'indicateur par les valeurs
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------+
//| La formation de la fin de la bougie                              |
//+------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- remplissons les tampons d'indicateur par les valeurs
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Remplissons du I-ème élément des tampons d'indicateur par les valeurs vides|
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- établissons la valeur vide à la cellule des tampons d'indicateur
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000-2015, MetaQuotes Software Corp.

ArrayRange

Rend le nombre d'éléments dans la dimension indiquée du tableau.

```
int  ArrayRange(
   const void&   array[],      // tableau pour le contrôle
   int           rank_index    // nombre de dimension
   );
```

Paramètres

array[]

[in] Le tableau contrôlé.

rank_index

[in] L'index de la dimension.

La valeur rendue

Le nombre d'éléments dans une dimension detableau choisie.

Note

Puisque les index commencent par le zéro, la quantité de mesures du tableau est plus sur une unité que l'index de la dernière mesure.

Exemple:

```
void OnStart()
  {
//--- créons un tableau quadridimensionnel
   double array[][5][2][4];
//--- spécifions la taille de la mesure de zéro
   ArrayResize(array,10,10);
//---imprimons les dimensions des mesures
   int temp;
   for(int i=0;i<4;i++)
     {
      //--- recevons la taille de la i-ième mesure
      temp=ArrayRange(array,i);
      //--- imprimons
      PrintFormat("dim = %d, range = %d",i,temp);
     }
//--- Le résultat
// dim = 0, range = 10
// dim = 1, range = 5
// dim = 2, range = 2
// dim = 3, range = 4
  }
```

© 2000-2015, MetaQuotes Software Corp.

ArrayResize

Met la nouvelle grandeur dans la première dimension du tableau

```
int  ArrayResize(
   void&  array[],              // tableau transmis selon la référence
   int    new_size,             // nouvelle grandeur du tableau
   int    reserve_size=0        // valeur de grandeur réservé (surabondant)
   );
```

Paramètres

array[]

[out] Le tableau pour le changement des grandeurs.

new_size

[in] Une nouvelle grandeur pour la première dimension.

reserve_size=0

[in] La grandeur pour la réserve supplémentaire.

La valeur rendue

A l'exécution réussi la fonction rend la quantité de tous les éléments se trouvant dans le tableau après le changement de la grandeur, autrement elle rend -1 et le tableau ne change pas la grandeur.

Note

La fonction peut être appliquée seulement vers les tableaux dynamiques. Il est nécessaire de tenir que c'est impossible de changer la taille des tableaux dynamiques fixés à titre des tampons d'indicateur par la fonction SetIndexBuffer(). Pour les tampons d'indicateur toutes les opérations du changement de la taille sont produites par le sous-système exécutant du terminal.

Общее число элементов в массиве не может превышать 2147483647.

A la distribution fréquente de la mémoire il est recommandé d'utiliser le troisième paramètre donnant la réserve pour la réduction du nombre de la distribution physique de la mémoire. Tous les appels ultérieurs de la fonction ArrayResize n'amènent pas à la redistribution physique de la mémoire, mais seulement la taille de la première dimension du tableau change au sein de la mémoire réservée. Il faut se rappeler que le troisième paramètre sera utilisé seulement quand il y aura une distribution physique de la mémoire, par exemple:

```
ArrayResize(arr,1000,1000);
for(int i=1;i<3000;i++)
   ArrayResize(arr,i,1000);
```

Dans ce cas il y aura 2 redistributions de la mémoire, une fois jusqu'à l'entrée dans la boucle de 2000 éléments, en cela la dimension du tableau sera fixé à 1000 et une deuxième fois - s'il i est égal à 2000. Si on va baisser le troisième paramètre, il y aura 2000 redistributions physiques de la mémoire et cela ralentira l'exécution du programme.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- compteurs
   ulong start=GetTickCount();
   ulong now;
   int   count=0;
//--- le tableau pour la démonstration de la variante rapide
   double arr[];
   ArrayResize(arr,100000,100000);
//--- vérifions la rapidité de la variante avec la réservation de la mémoire
   Print("--- Test Fast: ArrayResize(arr,100000,100000)");
   for(int i=1;i<=300000;i++)
     {
      //--- spécifions une nouvelle taille du tableau, ayant indiqué la réserve de 100
      ArrayResize(arr,i,100000);
      //--- à l'acquisition du chiffre rond déduisons la taille du tableau et le temps
      if(ArraySize(arr)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(arr)=%d Time=%d ms",count,ArraySize(arr),(now-star
         start=now;
        }
     }
//--- montrons maintenant, comment la variante sans réservation de la mémoire travaill
   double slow[];
   ArrayResize(slow,100000,100000);
//---
   count=0;
   start=GetTickCount();
   Print("---- Test Slow: ArrayResize(slow,100000)");
//---
   for(int i=1;i<=300000;i++)
     {
      //--- spécifions une nouvelle taille du tableau, mais sans une réserve supplémen
      ArrayResize(slow,i);
      //--- à l'acquisition du chiffre rond déduisons la taille du tableau et le temps
      if(ArraySize(slow)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(slow)=%d Time=%d ms",count,ArraySize(slow),(now-st
         start=now;
        }
     }
  }
//--- Le résultat approximatif de l'exécution du script
/*
   Test_ArrayResize (EURUSD,H1)   --- Test Fast: ArrayResize(arr,100000,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(arr)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(arr)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(arr)=300000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   ---- Test Slow: ArrayResize(slow,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(slow)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(slow)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(slow)=300000 Time=228511 ms
*/
```

© 2000-2015, MetaQuotes Software Corp.

Voir aussi

ArrayInitialize

© 2000-2015, MetaQuotes Software Corp.

ArraySetAsSeries

Établit le drapeau AS_SERIES à l'objet indiqué du tableau dynamique, l'indexation des éléments du tableau sera produite comme dans les séries temporelles.

```
bool  ArraySetAsSeries(
   const void&  array[],     // tableau selon la référence
   bool         flag         // true signifie l'ordre inverse de l'indexation
   );
```

Paramètres

array[]

[in][out] Le tableau numérique pour l'installation.

flag

[in] La direction de l'indexation du tableau.

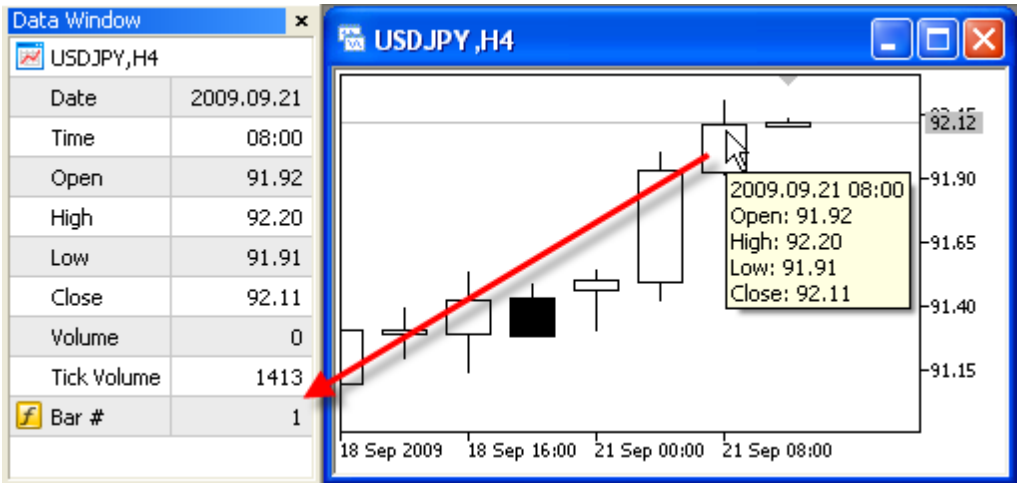
La valeur rendue

Rend true en cas du succès, autrement false.

Note

Le drapeau AS_SERIES ne peut pas être établi près des tableaux multidimensionnels et près des tableaux statiques (c'est-à-dire les tableaux, dont la grandeur dans les crochets est indiqué encore à l'étape de la compilation). L'indexation dans la série temporelle se distingue du tableau ordinaire par ce que l'indexation des éléments de la série temporelle est produite de la fin du tableau vers le début (de plus nouveau aux données les plus vieilles).

Exemple: l'indicateur montrant le numéro de la barre



© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Numeration
#property indicator_label1  "Numeration"
#property indicator_type1   DRAW_LINE
#property indicator_color1  CLR_NONE
//--- indicator buffers
double         NumerationBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,NumerationBuffer,INDICATOR_DATA);
//--- spécifions l'indexation pour le tampon comme à la série temporelle
   ArraySetAsSeries(NumerationBuffer,true);
//--- spécifions l'exactitude d'affichage à DataWindow
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//--- comment le nom de l'indicateur du tableau sera affiché dans DataWindow
   PlotIndexSetString(0,PLOT_LABEL,"Bar #");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---  conservons le temps de l'ouverture de la barre zéro actuelle
   static datetime currentBarTimeOpen=0;
//--- retournons l'accès au tableau time[] - faisons comme à la série temporelle
   ArraySetAsSeries(time,true);
//--- si le temps de la barre zéro se distingue de ce qu'on conserve
   if(currentBarTimeOpen!=time[0])
     {
     //--- numérotons tous les barres du moment actuel en profondeur le graphique
      for(int i=rates_total-1;i>=0;i--) NumerationBuffer[i]=i;
      currentBarTimeOpen=time[0];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

L'accès aux séries temporelles, ArrayGetAsSeries

© 2000-2015, MetaQuotes Software Corp.

ArraySize

Rend le nombre d'éléments dans le tableau.

```
int  ArraySize(
   const void&  array[]    // tableau contrôlé
   );
```

Paramètres

array[]

[in] Le tableau de n'importe quel type.

La valeur rendue

La valeur du type int.

Note

Pour le tableau unidimentionnel la valeur rendue par la fonction ArraySize,est égale à la valeur ArrayRange(array,0).

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
void OnStart()
  {
//--- la création des tableaux
   double one_dim[];
   double four_dim[][10][5][2];
//--- les tailles
   int one_dim_size=25;
   int reserve=20;
   int four_dim_size=5;
//--- la variable auxiliaire
   int size;
//--- allouons la mémoire sans réservation
   ArrayResize(one_dim,one_dim_size);
   ArrayResize(four_dim,four_dim_size);
//--- 1. un tableau unidimensionnel
   Print("+============================================================+");
   Print("Les tailles des tableaux:");
   Print("1. Un tableau unidimensionnel");
   size=ArraySize(one_dim);
   PrintFormat("La taille de la mesure nulle = %d, La taille du tableau = %d",one_dim_
//--- 2. un tableau multidimensionnel
   Print("2. Un tableau multidimensionnel");
   size=ArraySize(four_dim);
   PrintFormat("La taille de la mesure nulle = %d, La taille du tableau = %d",four_dim
//--- les tailles des mesures
   int d_1=ArrayRange(four_dim,1);
   int d_2=ArrayRange(four_dim,2);
   int d_3=ArrayRange(four_dim,3);
   Print("La vérification:");
   Print("La mesure nulle = La taille du tableau / (La première mesure * La deuxième m
   PrintFormat("%d = %d / (%d * %d * %d)",size/(d_1*d_2*d_3),size,d_1,d_2,d_3);
//--- 3. un tableau unidimensionnel avec la réservation de la mémoire
   Print("3. Un tableau unidimensionnel avec la réservation de la mémoire");
//--- augmentons la valeur à 2 fois
   one_dim_size*=2;
//--- allouons la mémoire avec la réservation
   ArrayResize(one_dim,one_dim_size,reserve);
//--- imprimons la taille
   size=ArraySize(one_dim);
   PrintFormat("La taille avec la réservation = %d, La taille réelle du tableau = %d",
  }
```

© 2000-2015, MetaQuotes Software Corp.

ArraySort

Сортирует многомерный числовой массив по возрастанию значений в первом измерении.

```
bool  ArraySort(
   void&  array[]      // tableau pour le triage
   );
```

Paramètres

array[]

[in][out] Le tableau numérique pour le triage.

La valeur rendue

Rend true en cas du succès, autrement false.

Note

Массив всегда сортируется по возрастанию, независимо от значения флага AS_SERIES.

Функции ArraySort и ArrayBSearch принимают в качестве параметра массив любой размерности, при этом сортировка и поиск происходят только по первому (нулевому) измерению.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
#property description "L'indicateur analyse les données pour le dernier mois et colore
#property description "et les grands volumes de tick. Pour la définition de telles bou
#property description "du tableau des volumes de tick. Les bougies dont les volumes fo
#property description " pour cents du tableau sont considérées comme petites. Les bouq
#property description "les derniers InpBigVolume pour cents du tableau, sont considéré
//--- les réglages de l'indicateur
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot
#property indicator_label1  "VolumeFactor"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrDodgerBlue,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- la constante prédéterminée
#define INDICATOR_EMPTY_VALUE 0.0
//---les paramètres d'entrée
input int InpSmallVolume=15; // Le pour-cent des petits volumes (<50)
input int InpBigVolume=20;   // Le pour-cent des grands volumes (<50)
//--- le temps du début de l'analyse (se déplacera)
datetime ExtStartTime;
//---les tampons d'indicateur
double   ExtOpenBuff[];
double   ExtHighBuff[];
double   ExtLowBuff[];
double   ExtCloseBuff[];
double   ExtColorBuff[];
//--- les valeurs limites des volumes pour afficher les bougies
long     ExtLeftBorder=0;
long     ExtRightBorder=0;
//+------------------------------------------------------------------+
//| La réception des significations des limites pour les volumes de tick
//+------------------------------------------------------------------+
bool GetVolumeBorders(void)
  {
//--- les variables
   datetime stop_time;  // le temps de la fin du copiage
   long     buff[];     // le tampon, où nous copions
//---le temps de la fin - le temps courant
   stop_time=TimeCurrent();
//--- l'heure de début - un mois plus tôt que le courant
   ExtStartTime=GetStartTime(stop_time);
//--- recevons les valeurs des volumes de tick
   ResetLastError();
   if(CopyTickVolume(Symbol(),Period(),ExtStartTime,stop_time,buff)==-1)
     {
      //--- on n' pas réussi de recevoir des données, rendons false pour l'exécution c
      PrintFormat("On n' pas réussi de recevoir les valeurs du volume de tick. Le code
      return(false);
     }
//---calculons la taille du tableau
   int size=ArraySize(buff);
//--- trions le tableau
   ArraySort(buff);
//--- définissons les valeurs des limites gauche et droite pour les volumes de tick
   ExtLeftBorder=buff[size*InpSmallVolume/100];
   ExtRightBorder=buff[(size-1)*(100-InpBigVolume)/100];
//---l'exécution réussie
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Obtenir la date qui est moins d'un mois que la date transmise    |
//+------------------------------------------------------------------+
datetime GetStartTime(const datetime stop_time)
  {
//--- transcrivons le temps de la fin vers la variable de la structure du type MqlDate
   MqlDateTime temp;
   TimeToStruct(stop_time,temp);
//--- recevons la date moins pour un mois
   if(temp.mon>1)
      temp.mon-=1;  // le mois courant n'est pas le premier en année, cela signifie le
   else
     {
      temp.mon=12;  // le mois courant est le premier en année, cela signifie que le m
      temp.year-=1; // et le numéro de l'année est moins pour un
     }
//---le nombre du jour sera pas plus 28
   if(temp.day>28)
      temp.day=28;
//--- rendons la date reçue
   return(StructToTime(temp));
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- la vérification si les paramètres d'entrée satisfont aux conditions
   if(InpSmallVolume<0 || InpSmallVolume>=50 || InpBigVolume<0 || InpBigVolume>=50)
     {
      Print("Les paramètres d'entrée invalides");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,ExtOpenBuff);
   SetIndexBuffer(1,ExtHighBuff);
   SetIndexBuffer(2,ExtLowBuff);
   SetIndexBuffer(3,ExtCloseBuff);
   SetIndexBuffer(4,ExtColorBuff,INDICATOR_COLOR_INDEX);
//--- définissons la valeur qui ne peut être affichée
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- spécifions l'inscription pour les tampons d'indicateur
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- la vérification s'il y a encore des barres non-traitées
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(prev_calculated<rates_total)
     {
      //--- la réception des nouvelles valeurs des limites gauche et droite pour les v
      if(!GetVolumeBorders())
         return(0);
     }
//--- la variable de début pour calculer les barres
   int start=prev_calculated;
//---si les valeurs de l'indicateur ont été déjà calculé sur un tick précédent, alors
   if(start>0)
      start--;
//--- établissons la direction directe de l'indexation dans les séries temporelles
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
   ArraySetAsSeries(tick_volume,false);
//--- le cycle du calcul des valeurs de l'indicateur
   for(int i=start;i<rates_total;i++)
     {
      //--- colorions les bougies à partir de la date de départ
      if(ExtStartTime<=time[i])
        {
         //--- si la valeur n'est pas moins de la limite droite, colorions la bougie
         if(tick_volume[i]>=ExtRightBorder)
           {
            //--- recevons les données pour le dessin de la bougie
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- la couleur DodgerBlue
            ExtColorBuff[i]=0;
            //--- continuons le cycle
            continue;
           }
         //---si la valeur n'est pas plus de la limite gauche, colorions la bougie
         if(tick_volume[i]<=ExtLeftBorder)
           {
            //---recevons les données pour le dessin de la bougie
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- la couleur Orange
            ExtColorBuff[i]=1;
            //--- continuons le cycle
            continue;
           }
        }
      //--- pour les barres qui ne sont pas prises en compte mettons la valeur vide
      ExtOpenBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtHighBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtLowBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtCloseBuff[i]=INDICATOR_EMPTY_VALUE;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Voir aussi

ArrayBsearch

© 2000-2015, MetaQuotes Software Corp.

La conversion des données

Le groupe des fonctions assurant la conversion des données d'un format aux données d'un autre format.

Il faut marquer spécialement la fonction NormalizeDouble(), qui assure l'exactitude demandée de la représentation du prix. Dans les opérations commerciales on ne peut pas utiliser les prix non normalisés, si l'exactitude dépasse les prix demandés par le serveur commercial même sur un signe.

Fonction	Action
CharToString	La conversion du code de symbole dans une chaîne d'un symbole
DoubleToString	La conversion de la valeur numérique à la chaîne de texte avec l'exactitude indiquée
EnumToString	La conversion de la valeur d'un transfert de n'importe quel type dans une chaîne
NormalizeDouble	L'arrondissement du nombre de la virgule flottante à l'exactitude indiquée
StringToDouble	La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type double
StringToInteger	La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type int
StringToTime	La conversion de la chaîne, contenant le temps et/ou la date dans le format "yyyy.mm.dd [hh:mi]", dans le nombre du type datetime
TimeToString	La conversion de la valeur, contenant le temps en secondes, passé après le 01.01.1970, à la chaîne du format "yyyy.mm.dd hh:mi"
IntegerToString	La conversion de la valeur du type entier à la chaîne de la longueur indiquée
ShortToString	La conversion du code de symbole (unicode)à la chaîne d'un symbole
ShortArrayToString	Copie la partie du tableau dans la chaîne
StringToShortArray	Copie la chaîne par un symbole à l'endroit spécifié du tableau du type ushort
CharArrayToString	Convertit le code de symbole (ansi) à la chaîne d'un symbole
StringToCharArray	Copie par un symbole la chaîne convertie de Unicode à ANSI à l'endroit spécifié du tableau du type uchar

© 2000-2015, MetaQuotes Software Corp.

ColorToARGB	Convertit le type color en type uint pour recevoir la représentation ARGB de la couleur.
ColorToString	Convertit la valeur de la couleur à la chaîne du type "R,G,B"
StringToColor	Convertit la chaîne du type "R,G,B" ou la chaîne, contenant le nom de la couleur, à la valeur du type color
StringFormat	Convertit le nombre à la chaîne conformément au format spécifié

Voir aussi

Utilisation de la page de code

© 2000-2015, MetaQuotes Software Corp.

CharToString

La conversion du code de symbole dans une chaîne d'un symbole

```
string  CharToString(
   uchar  char_code      // code numérique du symbole
   );
```

Paramètres

char_code

[in] Le code du symbole ANSI.

La valeur rendue

La chaîne, contenant le symbole ANSI.

© 2000-2015, MetaQuotes Software Corp.

CharArrayToString

Copie et convertit la partie du tableau du type uchar à la chaîne rendue.

```
string  CharArrayToString(
   uchar    array[],             // tableau
   int      start=0,             // position initiale dans le tableau
   int      count=-1             // nombre de symboles
   uint     codepage=CP_ACP      // page de code
   );
```

Paramètres

array[]

[in] Le tableau du type uchar.

start=0

[in] La position, par laquelle commence le copiage. Par défaut c'est 0.

count=-1

[in] Le nombre d'éléments de tableau pour le copiage. Définit la longueur de la chaîne de résultat. La valeur implicite est-1, qui signifie le copiage jusqu'à la fin du tableau ou jusqu'au 0 de terminal.

codepage=CP_ACP

[in] La valeur de la page de code. Pour les pages de code les plus utilisés on prévoit les constantes correspondantes.

La valeur rendue

La chaîne.

Voir aussi

Utilisation de la page de code

© 2000-2015, MetaQuotes Software Corp.

ColorToARGB

Convertit le type color en type uint pour recevoir la représentation ARGB de la couleur. Le format de la couleur ARGB est utilisé à la créationde la ressource graphique , de la sortie du texte et dans la classe de la bibliothèque standard CCanvas.

```
uint  ColorToARGB(
   color  clr,          // La couleur convertible dans le format color
   uchar  alpha=255     // Le canal alpha dirigeant la transparence de la couleur
   );
```

Réglages

clr

[in] La valeur de la couleur dans la variable du type color.

alpha

[in] La valeur du canal alpha, pour la réception de la couleur dans le formatARGB. Est spécifié par la valeur de 0 (la couleur du pixel appliqué ne change pas du tout l'affichage du pixel inférieur) jusqu'à 255 (la couleur est appliquée complètement et couvre la couleur du pixel inférieur). La transparence de la couleur en pourcentage est calculée comme (1-alpha/255) *100%, c'est-à-dire que plus la valeur du canal est moins grand plus la couleur est transparente.

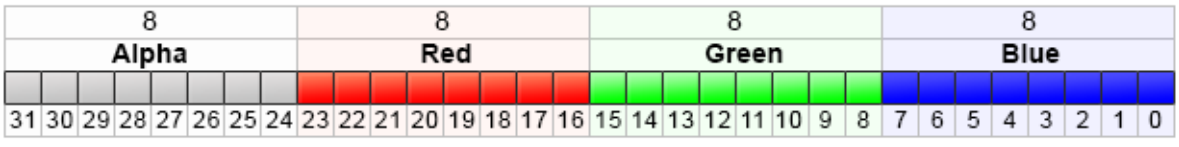
La valeur rendue

La représentation de la couleur dans le format ARGB, où dans quatre octets du type uint sont prescrit par ordre les valeurs Alfa, Red, Green, Blue (le canal alpha, rouge, vert, bleu clair).

Note

Le format principal utilisé partout pour la description de la couleur du pixel sur l'écran de l'écran en infographie est RGB, où selon les noms des couleurs de base sont spécifiés les composants rouge (Red), vert (Green) et bleu (Blue) de la couleur. Chaque composant est décrit par un octet qui spécifie la densité de la couleur donnée dans l'intervalle de 0 jusqu'à 255 (de 0x00 jusqu'à 0XFF dans le format hexadécimal). Puisque la couleur blanche contient toutes les couleurs, il est décrit comme comme 0xFFFFFF, c'est-à-dire chacun de trois composants est présenté par la valeur maximal 0xFF.

Mais dans un certain nombre des tâches on demande l'indication de la transparence de la couleur pour décrire, comment l'image va se présenter, si la couleur avec une certaine transparence est appliquée sur cette image. Pour tels cas on introduit la notion du canal alpha (Alpha), qui est introduit comme le composant supplémentaire au format RGB. Le schéma du format ARGB est montré sur le dessin.



Les valeurs ARGB sont indiquées d'habitude dans le format hexadécimal, où chaque paire des chiffres présente par ordre les valeurs des canaux Alpha, Red, Green et Blue. Par exemple, la couleur ARGB 80FFFF00 signifie la couleur jaune avec la transparence 50.2 %. Au début est 0x80, qui spécifie la valeur 50.2% du canal alpha, puisque c'est 50.2% de la valeur 0xFF; ensuite la première paire FF présente la valeur maximale du composant rouge; la paire suivante FF spécifie la même

© 2000-2015, MetaQuotes Software Corp.

force du composant vert; et la dernière paire 00 présente la valeur minimale du composant bleu (l'absence de la couleur bleue). L'addition de la couleur verte et rouge donne la couleur jaune. Si le canal alpha n'est pas utilisé, l'enregistrement peut être réduite jusqu'à 6 chiffres RRGGBB, c'est pourquoi les valeurs du canal alpha sont stockées dans les octets entiers supérieurs comme uint.

En fonction du contexte les chiffres qui contiennent 16 symboles peuvent être enregistrés avec le préfixe '0x' ou '#', par exemple, 80FFFF00, ou 0x80FFFF00, ou #80FFFF00.

Exemple:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- spécifions la transparence
   uchar alfa=0x55;  // la valeur 0x55 signifit 55/255=21.6 % de la transparence
   //--- déduisons la conversion au ARGB pour la couleur  clrBlue
   PrintFormat("0x%.8X - clrBlue",clrBlue);
   PrintFormat("0x%.8X - clrBlue ARGB with alfa=0x55 (transparency 21.6%%)",ColorToARG
   //---  déduisons la conversion au ARGB pour la couleur clrGreen
   PrintFormat("0x%.8X - clrGreen",clrGreen);
   PrintFormat("0x%.8X - clrGreen ARGB with alfa=0x55 (transparency 21.6%%)",ColorToAR
   //---  déduisons la conversion au ARGB pour la couleur  clrRed
   PrintFormat("0x%.8X - clrRed",clrRed);
   PrintFormat("0x%.8X - clrRed ARGB with alfa=0x55 (transparency 21.6%%)",ColorToARGB
  }
```

Voir aussi

Les ressources, ResourceCreate(), TextOut(), Le type color, Les types char, short, int et long

© 2000-2015, MetaQuotes Software Corp.

ColorToString

Convertit la valeur de la couleur à la chaîne du type "R,G,B".

```
string  ColorToString(
   color   color_value,     // valeur de la couleur
   bool    color_name       // montrez le nom de couleur ou non
   );
```

Paramètres

color_value

[in] La valeur de la couleur dans la variable du type color.

color_name

[in] Le signe de la nécessité de rendre le nom de la couleur, dans le cas où la valeur de la couleur coïncide avec une des constantes de couleur prédéterminées.

La valeur rendue

La représentation de chaîne de la couleur dans l'aspect "R,G,B", où R, G et B représentent les constantes décimales transcrites à la chaîne avec la valeur de 0 jusqu'à 255. Si on a spécifié le paramètre color_name=true, on fait la tentative de convertir la valeur de la couleur au nom de la couleur.

Exemple:

```
   string clr=ColorToString(C'0,255,0'); // couleur verte
   Print(clr);

   clr=ColorToString(C'0,255,0',true);   // recevoir la constante de couleur
   Print(clr);
```

© 2000-2015, MetaQuotes Software Corp.

DoubleToString

La conversion de la valeur numérique à la chaîne de texte.

```
string  DoubleToString(
   double  value,      // nombre
   int     digits=8    // le nombre de chiffres après le point
   );
```

Paramètres

value

[in] La valeur avec la virgule flottante.

digits

[in] Le format de l'exactitude. Si la valeur digits est dans le domaine de 0 jusqu'à 16, on recevra la représentation de chaîne du nombre avec le nombre indiqué des signes après la virgule. Si la valeur digits est dans le domaine de -1 jusqu'au -16, on recevra la représentation de chaîne du nombre dans le format scientifique avec le nombre indiqué des signes après la virgule. Dans tous les autres cas la valeur de chaîne contiendra les 8 chiffres après le point.

La valeur rendue

La chaîne, contenant la représentation symbolique du nombre dans le format indiqué de l'exactitude.

Exemple:

```
   Print("DoubleToString(120.0+M_PI) : ",DoubleToString(120.0+M_PI));
   Print("DoubleToString(120.0+M_PI,16) : ",DoubleToString(120.0+M_PI,16));
   Print("DoubleToString(120.0+M_PI,-16) : ",DoubleToString(120.0+M_PI,-16));
   Print("DoubleToString(120.0+M_PI,-1) : ",DoubleToString(120.0+M_PI,-1));
   Print("DoubleToString(120.0+M_PI,-20) : ",DoubleToString(120.0+M_PI,-20));
```

Voir aussi

NormalizeDouble, StringToDouble

© 2000-2015, MetaQuotes Software Corp.

EnumToString

La transformation de la valeur de l'énumération de n'importe quel type à la représentation de texte.

```
string  EnumToString(
   any_enum  value      // La valeur de l'énumération de tout type
   );
```

Paramètres

value

[in] La valeur de l'énumération de tout type.

La valeur rendue

Une chaîne contenant la représentation de texte de la valeur. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

La fonction peut définir dans la variable _LastError les valeurs suivantes des erreurs:

- ERR_INTERNAL_ERROR – l'erreur de l'environnement de l'exécution
- ERR_NOT_ENOUGH_MEMORY - pas assez de mémoire pour terminer l'opération
- ERR_INVALID_PARAMETER - Il est impossible de permettre le nom de la valeur de l'énumération

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
enum interval  // l'énumération des constantes nommées
  {
   month=1,     // l'intervalle dans un mois
   two_months,  // deux mois
   quarter,     // trois mois - le quartier
   halfyear=6,  // le semestre
   year=12,     // un an - 12 mois
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- établissons  l'intervalle temporaire qui est égal au mois
   interval period=month;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- établissons  l'intervalle temporaire qui est égal au quartier (trois mois))
   period=quarter;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- établissons  l'intervalle temporaire qui est égal à un an (12 mois))
   period=year;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- vérifions comment le type de l'ordre est déduit
   ENUM_ORDER_TYPE type=ORDER_TYPE_BUY;
   Print(EnumToString(type)+"="+IntegerToString(type));

//--- vérifions comment une valeur incorrecte est déduite
   type=WRONG_VALUE;
   Print(EnumToString(type)+"="+IntegerToString(type));

// Le résultat de l'exécution:
// month=1
// quarter=3
// year=12
// ORDER_TYPE_BUY=0
// ENUM_ORDER_TYPE::-1=-1
  }
```

Voir aussi

Les énumérations, Input les variables

© 2000-2015, MetaQuotes Software Corp.

IntegerToString

La conversion de la valeur du type entier à la chaîne de la longueur indiquée et rend la chaîne reçue.

```
string  IntegerToString(
   long     number,              // nombre
   int      str_len=0,           // longueur de la chaîne sur la sortie
   ushort   fill_symbol=' '      // remplisseur
   );
```

Paramètres

number

[in] Le nombre pour la conversion.

str_len=0

[in] La longueur de la chaîne. Si la longueur de la chaîne reçue sera plus grande que la longueur indiquée, la chaîne ne se coupe pas. Si la longueur de la chaîne reçue sera plus petite, la chaîne reçue sera complétée par le symbole-le remplisseur.

fill_symbol=' '

[in] Le symbole-le remplisseur. C'est un espace par défaut.

La valeur rendue

La chaîne.

© 2000-2015, MetaQuotes Software Corp.

ShortToString

La conversion du code de symbole (unicode)à la chaîne d'un symbole et rend la chaîne reçue.

```
string  ShortToString(
   ushort  symbol_code      // symbole
   );
```

Paramètres

symbol_code

[in] Le code de symbole. Au lieu du code du symbole on peut utiliser la chaîne littéral contenant le symbole, ou la chaîne avec le code hexadécimal de 2 bytes, correspondant au symbole du tableau Unicode.

La valeur rendue

La chaîne.

© 2000-2015, MetaQuotes Software Corp.

ShortArrayToString

Copie la partie du tableau dans la chaîne.

```
string  ShortArrayToString(
   ushort  array[],      // tableau
   int     start=0,      // position de départ dans le tableau
   int     count=-1      // nombre de symboles
   );
```

Paramètres

array[]

[in] Le tableau du type ushort (l'analogue du type wchar_t).

start=0

[in] La position, par laquelle commence le copiage. Par défaut c'est 0.

count=-1

[in] Le nombre d'éléments de tableau pour le copiage. Définit la longueur de la chaîne de résultat. La valeur implicite est-1, qui signifie le copiage jusqu'à la fin du tableau ou jusqu'au 0 de terminal.

La valeur rendue

La chaîne.

© 2000-2015, MetaQuotes Software Corp.

TimeToString

La conversion de la valeur, contenant le temps en secondes, passé après le 01.01.1970, à la chaîne du format "yyyy.mm.dd hh:mi"

```
string  TimeToString(
   datetime  value,                          // nombre
   int       mode=TIME_DATE|TIME_MINUTES     // format de la sortie
   );
```

Paramètres

value

[in] Le temps en secondes de 00:00 le 1 janvier 1970.

mode=TIME_DATE|TIME_MINUTES

[in] Le mode supplémentaire de la sortie de données. Peut être un drapeau ou le drapeau combiné:
TIME_DATE reçoit le résultat en forme de " yyyy.mm.dd " ,
TIME_MINUTES reçoit le résultat en forme de " hh:mm " ,
TIME_SECONDS reçoit le résultat en forme de " hh:mm:ss ".

La valeur rendue

La chaîne.

© 2000-2015, MetaQuotes Software Corp.

NormalizeDouble

L'arrondissement du nombre de la virgule flottante à l'exactitude indiquée.

```
double  NormalizeDouble(
   double   value,      // nombre normalisé
   int      digits      // nombre de chiffres après le point
   );
```

Paramètres

value

[in] La valeur avec la virgule flottante.

digits

[in] Le format de l'exactitude, le nombre de chiffres après le point décimal (0-8).

La valeur rendue

La valeur du type double avec l'exactitude donnée.

Note

Les valeurs calculées StopLoss, TakeProfit, et ainsi que les valeurs de prix de l'ouverture des ordres remis, doivent être normalisées avec l'exactitude, dont la valeur peut être reçue par la fonction Digits().

Il faut avoir en vue que le nombre normalisé quand l'on déduit au Journal à l'aide de Print () peut contenir plus de signes après la virgule, que vous attendez. Par exemple,

```
   double a=76.671;              // le nombre normalisé avec 3 signes après la virgule
   Print("Print(76.671)=",a);    // le déduisons comme il est
   Print("DoubleToString(a,8)=",DoubleToString(a,8)); // déduisons avec l'exactitude s
```

déduira dans le terminal:

```
   DoubleToString(a,8)=76.67100000
   Print(76.671)=76.67100000000001
```

Exemple:

```
   double pi=M_PI;
   Print("pi = ",DoubleToString(pi,16));

   double pi_3=NormalizeDouble(M_PI,3);
   Print("NormalizeDouble(pi,3) = ",DoubleToString(pi_3,16))
   ;
   double pi_8=NormalizeDouble(M_PI,8);
   Print("NormalizeDouble(pi,8) = ",DoubleToString(pi_8,16));

   double pi_0=NormalizeDouble(M_PI,0);
   Print("NormalizeDouble(pi,0) = ",DoubleToString(pi_0,16));
```

© 2000-2015, MetaQuotes Software Corp.

```
//   Résultat:
//   pi= 3.1415926535897931
//   NormalizeDouble(pi,3) = 3.1419999999999999
//   NormalizeDouble(pi,8) = 3.1415926499999998
//   NormalizeDouble(pi,0) = 3.0000000000000000
```

Voir aussi

DoubleToString, Les types matériels (double, float), La conformation des types,

© 2000-2015, MetaQuotes Software Corp.

StringToCharArray

Copie par un symbole la chaîne convertie de unicode à ansi à l'endroit spécifié du tableau du type uchar. La fonction rend le nombre d'éléments copiés.

```
int  StringToCharArray(
   string  text_string,          // chaîne-source
   uchar&  array[],              // tableau
   int     start=0,              // position de départ dans le tableau
   int     count=-1              // nombre de symboles
   uint    codepage=CP_ACP       // page de code
   );
```

Paramètres

text_string

[in] La chaîne pour le copiage.

array[]

[out] Le tableau du type uchar.

start=0

[in] La position, par laquelle commence le copiage. Par défaut c'est 0.

count=-1

[in] Le nombre d'éléments de tableau pour le copiage. Définit la longueur de la chaîne de résultat. La valeur implicite est-1, qui signifie le copiage jusqu'à la fin du tableau ou jusqu'au 0 de terminal. 0 de terminal sera aussi copié au tableau-récepteur, dans ce cas-là la grandeur du tableau dynamique peut être augmentée en cas de nécessité à la grandeur de la chaîne. Si la grandeur du tableau dynamique est plus grand que la longueur de la chaîne, la grandeur du tableau ne sera pas diminué.

codepage=CP_ACP

[in] La valeur de la page de code. Pour les pages de code les plus utilisés on prévoit les constantes correspondantes.

La valeur rendue

Le nombre d'éléments copiés.

Voir aussi

Utilisation de la page de code

© 2000-2015, MetaQuotes Software Corp.

StringToColor

Convertit la chaîne du type "R,G,B" ou la chaîne, contenant le nom de la couleur, à la valeur du type color.

```
color  StringToColor(
   string  color_string      // représentation de la chaîne de couleur
   );
```

Paramètres

color_string

[in] La représentation de chaîne de la couleur comme "R,G,B" ou le nom d'un des couleurs Web prédéterminés.

La valeur rendue

La valeur de la couleur.

Exemple:

```
   color str_color=StringToColor("0,127,0");
   Print(str_color);
   Print((string)str_color);
//--- on change la couleur un peu
   str_color=StringToColor("0,128,0");
   Print(str_color);
   Print((string)str_color);
```

© 2000-2015, MetaQuotes Software Corp.

StringToDouble

La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type double.

```
double  StringToDouble(
   string  value      // chaîne
   );
```

Paramètres

value

[in] La chaîne, contenant la représentation symbolique du nombre.

La valeur rendue

La valeur du type double.

© 2000-2015, MetaQuotes Software Corp.

StringToInteger

La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type int (entier).

```
long  StringToInteger(
   string  value      // chaîne
   );
```

Paramètres

value

[in] La chaîne, contenant le nombre.

La valeur rendue

La valeur du type long.

© 2000-2015, MetaQuotes Software Corp.

StringToShortArray

Copie la chaîne par un symbole à l'endroit spécifié du tableau du type ushort ushort. La fonction rend le nombre d'éléments copiés.

```
int  StringToShortArray(
   string  text_string,      // chaîne-source
   ushort& array[],          // tableau
   int     start=0,          // position initiale dans le tableau
   int     count=-1          // nombre de symboles
   );
```

Paramètres

text_string

[in] La chaîne pour le copiage.

array[]

[out] Le tableau du type ushort (analogue du type wchar_t).

start=0

[in] La position, par laquelle commence le copiage. Par défaut c'est 0.

count=-1

[in] Le nombre d'éléments de tableau pour le copiage. Définit la longueur de la chaîne de résultat. La valeur implicite est-1, qui signifie le copiage jusqu'à la fin du tableau ou jusqu'au 0 de terminal. 0 de terminal sera aussi copié au tableau-récepteur, dans ce cas-là la grandeur du tableau dynamique peut être augmentée en cas de nécessité à la grandeur de la chaîne. Si la grandeur du tableau dynamique est plus grand que la longueur de la chaîne, la grandeur du tableau ne sera pas diminué.

La valeur rendue

Le nombre d'éléments copiés.

© 2000-2015, MetaQuotes Software Corp.

StringToTime

La conversion de la chaîne, contenant le temps et/ou la date dans le format "yyyy.mm.dd [hh:mi]", dans le nombre du type datetime.

```
datetime  StringToTime(
   string  value      // chaîne-date
   );
```

Paramètres

value

[in] La chaîne au format " yyyy.mm.dd hh:mi ".

La valeur rendue

La valeur du type datetime, contenant le nombre de secondes qui sont passées depuis le 01.01.1970.

© 2000-2015, MetaQuotes Software Corp.

StringFormat

Formate les paramètres reçus et rend la chaîne.

```
string  StringFormat(
   string  format,     // chaîne avec la description du format
   ...     ...         // paramètres
   );
```

Paramètres

format

[in] La chaîne, contenant le moyen du formatage. Les règles du formatage sont les mêmes, comme pour la fonction PrintFormat

...

[in] Les paramètres divisés par la virgule.

La valeur rendue

La chaîne.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
void OnStart()
  {
//--- строковые переменные
   string output_string;
   string temp_string;
   string format_string;
//--- подготовим заголовок спецификации
   temp_string=StringFormat("Спецификация контракта для %s:\n",_Symbol);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int digits=(int)SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   temp_string=StringFormat("   SYMBOL_DIGITS = %d (количество знаков после запятой)\n
                            digits);
   StringAdd(output_string,temp_string);
//--- вывод значения double с переменным количеством цифр после десятичной точки
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   format_string=StringFormat("   SYMBOL_POINT = %%.%df (значение одного пункта)\n",
                              digits);
   temp_string=StringFormat(format_string,point_value);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   temp_string=StringFormat("   SYMBOL_SPREAD = %d (текущий спред в пунктах)\n",
                            spread);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int min_stop=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_STOPS_LEVEL = %d (минимальный отступ в пу
                            min_stop);
   StringAdd(output_string,temp_string);
//--- вывод значения double без дробной части
   double contract_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_CONTRACT_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_CONTRACT_SIZE = %.f (размер контракта)\n"
                            contract_size);
   StringAdd(output_string,temp_string);
//--- вывод значения double с точностью по умолчанию
   double tick_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_TICK_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_TICK_SIZE = %f (минимальное изменение це
                            tick_size);
   StringAdd(output_string,temp_string);
//--- определение способа начисления свопов
   int swap_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_SWAP_MODE);
   string str_swap_mode;
   switch(swap_mode)
     {
      case SYMBOL_SWAP_MODE_DISABLED: str_swap_mode="SYMBOL_SWAP_MODE_DISABLED (нет св
      case SYMBOL_SWAP_MODE_POINTS: str_swap_mode="SYMBOL_SWAP_MODE_POINTS (в пунктах)
      case SYMBOL_SWAP_MODE_CURRENCY_SYMBOL: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_
      case SYMBOL_SWAP_MODE_CURRENCY_MARGIN: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_
      case SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY
      case SYMBOL_SWAP_MODE_INTEREST_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST
      case SYMBOL_SWAP_MODE_INTEREST_OPEN: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST_OP
      case SYMBOL_SWAP_MODE_REOPEN_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_CUR
      case SYMBOL_SWAP_MODE_REOPEN_BID: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_BID (пе
     }
//--- вывод значения string
   temp_string=StringFormat("   SYMBOL_SWAP_MODE = %s\n",
                            str_swap_mode);
   StringAdd(output_string,temp_string);
//--- вывод значения double с точностью по умолчанию
   double swap_long=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_LONG);
```

© 2000-2015, MetaQuotes Software Corp.

```
   temp_string=StringFormat("   SYMBOL_SWAP_LONG = %f (своп на покупку)\n",
                            swap_long);
   StringAdd(output_string,temp_string);
//--- вывод значения double с точностью по умолчанию
   double swap_short=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_SHORT);
   temp_string=StringFormat("   SYMBOL_SWAP_SHORT = %f (своп на продажу)\n",
                            swap_short);
   StringAdd(output_string,temp_string);
//--- определение режима торговли
   int trade_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_MODE);
   string str_trade_mode;
   switch(trade_mode)
     {
      case SYMBOL_TRADE_MODE_DISABLED: str_trade_mode="SYMBOL_TRADE_MODE_DISABLED (тор
      case SYMBOL_TRADE_MODE_LONGONLY: str_trade_mode="SYMBOL_TRADE_MODE_LONGONLY (pas
      case SYMBOL_TRADE_MODE_SHORTONLY: str_trade_mode="SYMBOL_TRADE_MODE_SHORTONLY (p
      case SYMBOL_TRADE_MODE_CLOSEONLY: str_trade_mode="SYMBOL_TRADE_MODE_CLOSEONLY (p
      case SYMBOL_TRADE_MODE_FULL: str_trade_mode="SYMBOL_TRADE_MODE_FULL (нет огранич
     }
//--- вывод значения string
   temp_string=StringFormat("   SYMBOL_TRADE_MODE = %s\n",
                            str_trade_mode);
   StringAdd(output_string,temp_string);
//--- вывод значения double в компактном виде
   double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   temp_string=StringFormat("   SYMBOL_VOLUME_MIN = %g (минимальный объем сделки)\n",v
   StringAdd(output_string,temp_string);
//--- вывод значения double в компактном виде
   double volume_step=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_STEP);
   temp_string=StringFormat("   SYMBOL_VOLUME_STEP = %g (минимальный шаг изменения об
   StringAdd(output_string,temp_string);
//--- вывод значения double в компактном виде
   double volume_max=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MAX);
   temp_string=StringFormat("   SYMBOL_VOLUME_MAX = %g (максимальный объем сделки)\n",
   StringAdd(output_string,temp_string);
//--- определение способа вычисления величины залоговых средств
   int calc_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_CALC_MODE);
   string str_calc_mode;
   switch(calc_mode)
     {
      case SYMBOL_CALC_MODE_FOREX:str_calc_mode="SYMBOL_CALC_MODE_FOREX (Forex)";break
      case SYMBOL_CALC_MODE_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_FUTURES (фьючерсы)
      case SYMBOL_CALC_MODE_CFD:str_calc_mode="SYMBOL_CALC_MODE_CFD (CFD)";break;
      case SYMBOL_CALC_MODE_CFDINDEX:str_calc_mode="SYMBOL_CALC_MODE_CFDINDEX (CFD на
      case SYMBOL_CALC_MODE_CFDLEVERAGE:str_calc_mode="SYMBOL_CALC_MODE_CFDLEVERAGE (C
      case SYMBOL_CALC_MODE_EXCH_STOCKS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_STOCKS (т
      case SYMBOL_CALC_MODE_EXCH_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FUTURES
      case SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FU
     }
//--- вывод значения string
   temp_string=StringFormat("   SYMBOL_TRADE_CALC_MODE = %s\n",
                            str_calc_mode);
   StringAdd(output_string,temp_string);
//--- вывод значения double с 2 цифрами после десятичной точки
   double margin_initial=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_INITIAL);
   temp_string=StringFormat("   SYMBOL_MARGIN_INITIAL = %.2f (начальная маржа)\n",
                            margin_initial);
   StringAdd(output_string,temp_string);
//--- вывод значения double с 2 цифрами после десятичной точки
   double margin_maintenance=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_MAINTENANCE);
   temp_string=StringFormat("   SYMBOL_MARGIN_MAINTENANCE = %.2f (поддерживающая маржа
```

© 2000-2015, MetaQuotes Software Corp.

```
                              margin_maintenance);
   StringAdd(output_string,temp_string);
//--- вывод значения int
   int freeze_level=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_FREEZE_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_FREEZE_LEVEL = %d (дистанция заморозки ор
                              freeze_level);
   StringAdd(output_string,temp_string);
   Print(output_string);
   Comment(output_string);
/* результат выполнения
   Спецификация контракта для EURUSD:
     SYMBOL_DIGITS = 5 (количество знаков после запятой)
     SYMBOL_POINT = 0.00001 (значение одного пункта)
     SYMBOL_SPREAD = 10 (текущий спред в пунктах)
     SYMBOL_TRADE_STOPS_LEVEL = 18 (минимальный отступ в пунктах для стоп-ордеров)
     SYMBOL_TRADE_CONTRACT_SIZE = 100000 (размер контракта)
     SYMBOL_TRADE_TICK_SIZE = 0.000010 (минимальное изменение цены)
     SYMBOL_SWAP_MODE = SYMBOL_SWAP_MODE_POINTS (в пунктах)
     SYMBOL_SWAP_LONG = -0.700000 (своп на покупку)
     SYMBOL_SWAP_SHORT = -1.000000 (своп на продажу)
     SYMBOL_TRADE_MODE = SYMBOL_TRADE_MODE_FULL (нет ограничений на торговые операции)
     SYMBOL_VOLUME_MIN = 0.01 (минимальный объем сделки)
     SYMBOL_VOLUME_STEP = 0.01 (минимальный шаг изменения объема сделки)
     SYMBOL_VOLUME_MAX = 500 (максимальный объем сделки)
     SYMBOL_TRADE_CALC_MODE = SYMBOL_CALC_MODE_FOREX (Forex)
     SYMBOL_MARGIN_INITIAL = 0.00 (начальная маржа)
     SYMBOL_MARGIN_MAINTENANCE = 0.00 (поддерживающая маржа)
     SYMBOL_TRADE_FREEZE_LEVEL = 0 (дистанция заморозки операций в пунктах)
*/
  }
```

Voir aussi

PrintFormat, DoubleToString,ColorToString, TimeToString

© 2000-2015, MetaQuotes Software Corp.

Les fonctions mathématiques

Un ensemble des fonctions mathématiques et trigonométriques.

Fonction	Action
MathAbs	Rend la valeur absolue (la valeur selon le module) du nombre transmis à elle
MathArccos	Rend la valeur d'arc cosinus x aux radians
MathArcsin	Rend la valeur d'arc sinus x aux radians
MathArctan	Rend l'arc tangent x aux radians
MathCeil	Rend la valeur entière numérique la plus proche par dessus
MathCos	Rend le cosinus du nombre
MathExp	Rend l'exposant au nombre
MathFloor	Rend la valeur entière numérique la plus proche par dessous
MathLog	Rend le logarithme naturel
MathLog10	Rend le logarithme du nombre selon le fondement 10
MathMax	Rend la valeur maximal des deux valeurs numériques
MathMin	Rend la valeur minimal des deux valeurs numériques
MathMod	Rend le reste réel après la division de deux nombres
MathPow	Elève le fondement à la puissance indiquée
MathRand	Rend le nombre entier pseudo-accidentelle dans le domaine de 0 jusqu'à 32767
MathRound	Arrondit le nombre jusqu'à l'entier plus proche
MathSin	Rend le sinus du nombre
MathSqrt	Rend la racine carrée
MathSrand	Établit l'état initial du générateur des nombres entiers quasi-aléatoires
MathTan	Rend la tangent du nombre
MathIsValidNumber	Vérifie l'exactitude d'un nombre réel

© 2000-2015, MetaQuotes Software Corp.

MathAbs

Rend la valeur absolue (la valeur selon le module) du nombre transmis à elle.

```
double  MathAbs(
   double  value      // nombre
   );
```

Paramètres

value

[in] La valeur numérique.

La valeur rendue

La valeur du type double, est plus que ou égal à zéro.

Note

On peut utiliser la fonction fabs() au lieu de la fonction MathAbs().

© 2000-2015, MetaQuotes Software Corp.

MathArccos

Rend la valeur d'arc cosinus x dans le domaine 0 au π dans les radians.

```
double  MathArccos(
   double  val     // -1<val<1
   );
```

Paramètres

val

[in] La valeur val est entre -1 et 1, dont l'arc cosinus doit être calculé.

La valeur rendue

L'arc cosinus du nombre est dans les radians. Si val est moins que -1 ou plus que 1, la fonction rend NaN (une valeur indéterminée).

Note

On peut utiliser la fonction acos() au lieu de la fonction MathArccos().

Voir aussi

Les types réels (double, float)

© 2000-2015, MetaQuotes Software Corp.

MathArcsin

Rend l'arc sinus x dans le domaine de -π/2 jusqu'au π/2 des radians.

```
double  MathArcsin(
   double  val      // -1<value<1
   );
```

Paramètres

val

[in] La valeur val est entre -1 et 1, dont l'arc sinus doit être calculé.

La valeur rendue

L'arc sinus du nombre val dans les radians dans le domaine de -π/2 jusqu'au π/2 des radians. Si val est moins que -1 ou plus que 1, la fonction rend NaN (une valeur indéterminée).

Note

On peut utiliser la fonction asin() au lieu de la fonction MathArcsin().

Voir aussi

Les types réels (double, float)

© 2000-2015, MetaQuotes Software Corp.

MathArctan

Rend l'arc tangent x. Si x est égal au 0, la fonction rend 0.

```
double  MathArctan(
   double  value      // tangent
   );
```

Paramètres

value

[in] Le nombre qui représente la tangente.

La valeur rendue

MathArctan rend la valeur dans le domaine de -π/2 jusqu'au les radians π/2.

Note

On peut utiliser la fonction atan() au lieu de la fonction MathArctan().

© 2000-2015, MetaQuotes Software Corp.

MathCeil

Rend la valeur entière numérique la plus proche par dessus.

```
double  MathCeil(
   double  val      // nombre
   );
```

Paramètres

val

[in] La valeur numérique.

La valeur rendue

La valeur numérique, représentant le plus petit nombre entier, qui est plus ou égal à val.

Note

On peut utiliser la fonction ceil() au lieu de la fonction MathCeil().

© 2000-2015, MetaQuotes Software Corp.

MathCos

La fonction rend le cosinus du nombre.

```
double  MathCos(
   double  value      // nombre
   );
```

Paramètres

value

[in] L'angle est aux radians.

La valeur rendue

La valeur du type double dans le domine de -1 jusqu'au 1.

Note

On peut utiliser la fonction cos()au lieu de la fonction MathCos().

© 2000-2015, MetaQuotes Software Corp.

MathExp

La fonction revient la valeur du nombre e à la puissance d.

```
double  MathExp(
   double  value      // puissance pour le nombre e
   );
```

Paramètres

value

[in] Le nombre définissant la puissance.

La valeur rendue

Le nombre du type double. Au débordement, la fonction rend INF (l'infinité), en cas de la perte de l'ordre de MathExp rend 0.

Note

On peut utiliser la fonction exp()au lieu de la fonction MathExp().

Voir aussi

Les types réels (double, float)

© 2000-2015, MetaQuotes Software Corp.

MathFloor

Rend la valeur entière numérique la plus proche par dessous.

```
double  MathFloor(
   double  val     // nombre
   );
```

Paramètres

val

[in] La valeur numérique.

La valeur rendue

La valeur numérique, représentant le plus grand nombre entier, qui est plus ou égal à val.

Note

On peut utiliser la fonction floor() au lieu de la fonction MathFloor().

© 2000-2015, MetaQuotes Software Corp.

MathLog

Rend le logarithme naturel.

```
double  MathLog(
   double  val      // nombre pour prendre le logarithme
   );
```

Paramètres

val

[in] La valeur dont le logarithme doit être calculé.

La valeur rendue

Le logarithme naturel est val en cas du succès. Si la valeur val est negative, la fonction rend NaN (une valeur indéterminée). Si val est égal au 0, la fonction rend INF (l'infinité).

Note

On peut utiliser la fonction log() au lieu de la fonction MathLog().

Voir aussi

Les types réels (double, float)

© 2000-2015, MetaQuotes Software Corp.

MathLog

Rend le logarithme du nombre selon le fondement 10.

```
double  MathLog10(
   double  val      // nombre pour prendre le logarithme
   );
```

Paramètres

val

[in] La valeur dont le logarithme décimal doit être calculé.

La valeur rendue

le logarithme décimal est val en cas du succès. Si la valeur val est négative, la fonction rend NaN (une valeur indéterminée). Si val est égal au 0, la fonction rend INF (l'infinité).

Note

On peut utiliser la fonction log10() au lieu de la fonction MathLog10().

Voir aussi

Les types réels(double, float)

© 2000-2015, MetaQuotes Software Corp.

MathMax

Rend la valeur maximal des deux valeurs numériques.

```
double  MathMax(
   double  value1,      // première valeur
   double  value2       // deuxième valeur
   );
```

Paramètres

value1

[in] La première valeur numérique.

value2

[in] La deuxième valeur numérique.

La valeur rendue

Le plus grand de deux nombres.

Note

On peut utiliser la fonction fmax()au lieu de la fonction MathMax(). Les fonctions fmax(), fmin(), MathMax(), MathMin() peuvent travailler avec les types entiers sans transformation au type double.

Si on transmet les paramètres des types différents dans la fonction, le paramètre du type bas est amené_ automatiquement au type principal. Le type de la valeur rendue correspond au type principal.

Si on transmet les données d'un type, aucune transformation n'est pas produite.

© 2000-2015, MetaQuotes Software Corp.

MathMin

Rend la valeur minimal des deux valeurs numériques.

```
double  MathMin(
   double  value1,      // première valeur
   double  value2       // deuxième valeur
   );
```

Paramètres

value1

[in] La première valeur numérique.

value2

[in] La deuxième valeur numérique.

La valeur rendue

Le plus petit de deux nombres.

Note

On peut utiliser la fonction fmin()au lieu de la fonction MathMin(). Les fonctions fmax(), fmin(), MathMax(), MathMin() peuvent travailler avec les types entiers sans transformation au type double.

Si on transmet les paramètres des types différents dans la fonction, le paramètre du type bas est amené_ automatiquement au type principal. Le type de la valeur rendue correspond au type principal.

Si on transmet les données d'un type, aucune transformation n'est pas produite.

© 2000-2015, MetaQuotes Software Corp.

MathMod

Rend le reste réel après la division de deux nombres.

```
double  MathMod(
   double  value,       // dividende
   double  value2       // diviseur
   );
```

Paramètres

value

[in] La valeur du dividence.

value2

[in] La valeur du diviseur.

La valeur rendue

La fonction MathMod calcule le reste réel f de val / y de manière que val = i * y + f , où i est le nombre entier, f a le même signe que val, et la valeur absolue f est moins que la valeur absolue y.

Note

On peut utiliser la fonction fmod()au lieu de la fonction MathMod().

© 2000-2015, MetaQuotes Software Corp.

MathPow

Elève le fondement à la puissance indiquée.

```
double  MathPow(
   double  base,          // fondement
   double  exponent       // valeur d'exposant
   );
```

Paramètres

base

[in] Le fondement.

exponent

[in] La valeur de la puissance.

La valeur rendue

La valeur du fondement, élevé à la puissance indiquée.

Note

On peut utiliser la fonction pow()au lieu de la fonction MathPow().

© 2000-2015, MetaQuotes Software Corp.

MathRand

Rend le nombre entier pseudo-accidentelle dans le domaine de 0 jusqu'à 32767.

```
int  MathRand();
```

La valeur rendue

Le nombre entier dans le domaine de 0 jusqu'au 32767.

Note

Avant le premier appel de la fonction il est nécessaire d'utiliser la fonction MathSrand, pour transmettre le générateur des nombres pseudo-accidentels à l'état initial.

Note

On peut utiliser la fonction rand()au lieu de la fonction MathRand().

© 2000-2015, MetaQuotes Software Corp.

MathRound

Rend la valeur arrondie jusqu'au nombre entier plus proche de la valeur numérique indiquée.

```
double  MathRound(
   double  value      // valeur arrondit
   );
```

Paramètres

value

[in] La valeur numérique pour l'arrondissement.

La valeur rendue

La valeur arrondie jusqu'à le nombre entier le plus proche.

Note

On peut utiliser la fonction round() au lieu de la fonction MathRound().

© 2000-2015, MetaQuotes Software Corp.

MathSin

Rend le sinus de l'angle indiqué.

```
double  MathSin(
   double  value      // argument dans les radians
   );
```

Paramètres

value

[in] L'angle dans les radians.

La valeur rendue

Le sinus de l'angle, indiqués dans les radians. Rend la valeur dans le domaine de -1 jusqu'au 1.

Note

On peut utiliser la fonction sin()au lieu de la fonction MathSin().

© 2000-2015, MetaQuotes Software Corp.

MathSqrt

Rend la racine carrée du nombre.

```
double  MathSqrt(
   double  value      // nombre positif
   );
```

Paramètres

value

[in] La valeur positive numérique.

La valeur rendue

La racine carrée du nombre value. Si value est négative, MathSqrt rend NaN (valeur indéterminée).

Note

On peut utiliser la fonction sqrt()au lieu de la fonction MathSqrt().

Voir aussi

Les types réels (double, float)

© 2000-2015, MetaQuotes Software Corp.

MathSrand

Établit l'état initial de la génération des nombres entiers quasi-aléatoires.

```
void  MathSrand(
   int  seed      // nombre initialisant
   );
```

Paramètres

seed

[in] Le nombre initial pour une série de nombres accidentels.

La valeur rendue

Il n'y a pas de valeur rendue.

Note

La fonction MathRand() est destinée à la génération de la succession des nombres quasi-aléatoires. L' appel MathSrand() avec le nombre défini initialisant permet de recevoir toujours la même succession des nombres quasi-aléatoires.

Pour la réception garantie de la succession qui ne se répéte pas utilisez l'appel MathSrand(GetTickCount()), puisque la valeurGetTickCount()augmente dès le moment du lancement du système d'exploitation et ne se répète pas pendant 49 jours, jusqu'au le moment quand le compteur inséré des millisecondes ne débordera pas. L'utilisation MathSrand(TimeCurrent()) ne convient pas patce que la fonctionTimeCurrent()rend le temps de l'arrivée du dernier tick, qui ne peut ne pas être changé longtemps, par exemple, aux jours fériés.

Il est mieux de faire l'initialisation du générateur des nombres quasi-aléatoires à l'aide de MathSrand () pour les indicateurs et les experts dans le gestionnaire OnInit (), cela délivrera des lancements suivants multiples du générateur dans OnTick() et OnCalculate().

Au lieu de la fonction MathSrand() on peut utiliser la fonctionsrand().

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
#property description "L'indicateur présente le théorème central limite, qui déclare:"
#property description "La somme du nombre assez grand de variables aléatoires faibleme
#property description "ayant approximativement les mêmes échelles (ou nul opérande de
#property description "ne contribue pas le dépôt définissant à la somme), a la distrib

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- les propriétés de la construction graphique
#property indicator_label1  "Label"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRoyalBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  5
//--- variable input
input int      sample_number=10;
//--- le tampon d'indicateur pour le dessin de la distribution
double         LabelBuffer[];
//--- le compteur des ticks
double         ticks_counter;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- le liage du tableau et du tampon d'indicateur
   SetIndexBuffer(0,LabelBuffer,INDICATOR_DATA);
//--- déploierons le tampon d'indicateur du présent au passé
   ArraySetAsSeries(LabelBuffer,true);
//--- initialisons le générateur des nombres aléatoires
   MathSrand(GetTickCount());
//--- initialisons le compteur des ticks
   ticks_counter=0;
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le tampon d'indicateur sera remis à zéro au compteur nul
   if(ticks_counter==0) ArrayInitialize(LabelBuffer,0);
//--- augmentons le compteur
   ticks_counter++;
//--- il faut périodiquement enlever le compteur des ticks pour ranimer la distributio
   if(ticks_counter>100)
     {
      Print("on a remit à zéro les valeurs de l'indicateur, commencerons à remplir les
      ticks_counter=0;
     }
//--- recevrons un échantillon de valeurs aléatoires en tant que la somme de trois nor
   for(int i=0;i<sample_number;i++)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //---le calcul de l'index de la cellule, où tombera un nombre aléatoire comme la
      int rand_index=0;
      //--- recevrons trois nombres aléatoires de 0 jusqu'à 7
      for(int k=0;k<3;k++)
        {
         //--- le reste de la division sur 7 rendra la valeur de 0 jusqu'à 6
         rand_index+=MathRand()%7;
        }
      //--- augmentons sur une unité la valeur dans la cellule avec le numéro rand_ind
      LabelBuffer[rand_index]++;
     }
//--- la sortie du gestionnaire OnCalculate()
   return(rates_total);
  }
```

© 2000-2015, MetaQuotes Software Corp.

MathTan

Rend la tangent du nombre.

```
double  MathTan(
   double  rad      // argument dans les radians
   );
```

Paramètres

rad

[in] L'angle dans les radians.

La valeur rendue

La tangent du nombre rad. Si rad est plus ou égal au 263 ou moins que ou égal au -263, on perd la valeur et la fonction rend un nombre indéterminé.

Note

On peut utiliser la fonction tan()au lieu de la fonction MathTan().

Voir aussi

Les types réels (double, float)

© 2000-2015, MetaQuotes Software Corp.

MathIsValidNumber

Vérifie l'exactitude d'un nombre réel

```
bool  MathIsValidNumber(
   double  number      // nombre pour vérifier
   );
```

Paramètres

number

[in] Le nombre vérifié.

La valeur rendue

Rend true, si la valeur vérifiée est le nombre réel admissible. Si la valeur vérifiée est un plus ou le moins l'infinité, ou "pas un nombre" (NaN - not a number), la fonction rend false.

Exemple:

```
   double abnormal=MathArcsin(2.0);
   if(!MathIsValidNumber(abnormal)) Print("Attention! MathArcsin(2.0) = ",abnormal);
```

Voir aussi

Les types réels (double, float)

© 2000-2015, MetaQuotes Software Corp.

Fonctions de chaîne

Le groupe des fonctions destinées au travail avec les données du type string.

Fonction	Action
StringAdd	Ajoute la chaîne selon la place indiquée par la sous-chaîne
StringBufferLen	Rend la grandeur du buffer distribué pour la chaîne.
StringCompare	Compare deux chaînes et rend 1 si la première chaîne est plus que deuxième; 0 - si les chaînes sont égales; -1 (moins un) - si la première chaîne est moins que la deuxième
StringConcatenate	Forme la chaîne des paramètres transmis
StringFill	Remplit la chaîne indiquée selon la place par les symboles indiqués
StringFind	La recherche de la sous-chaîne dans la chaîne
StringGetCharacter	Rend la valeur du symbole disposé dans la position indiquée de la chaîne
StringInit	Initialise la chaîne par les symboles indiqués et assure la grandeur indiquée de la chaîne
StringLen	Rend le nombre de symboles dans la chaîne
StringReplace	Remplace tous les sous-chaînes recherchés dans la chaîne pour la séquence spécifiée des caractères.
StringSetCharacter	Rend la copie de la chaîne avec la valeur changée du symbole dans la position indiquée
StringSplit	Reçoit les sous-chaînes de la chaîne indiquée par un séparateur spécifié et rend le montant des sous-chaînes reçus
StringSubstr	Extrait la sous-chaîne de la chaîne de texte qui commence de la position indiquée
StringToLower	Transforme tous les symboles de la chaîne indiquée aux symboles minuscules selon la place
StringToUpper	Transforme tous les symboles de la chaîne indiquée aux symboles majuscules selon la place
StringTrimLeft	Supprime les symboles de la conversion de chariot, les espaces et les symboles de la tabulation du début de la chaîne
StringTrimRight	Supprime les symboles de la conversion de

© 2000-2015, MetaQuotes Software Corp.

	chariot, les espaces et les symboles de la tabulation à la fin de la chaîne

© 2000-2015, MetaQuotes Software Corp.

StringAdd

Ajoute la chaîne selon la place indiquée par la sous-chaîne.

```
bool  StringAdd(
   string&  string_var,        // chaîne, à qui est ajouté
   string   add_substring      // chaîne ajoutée
   );
```

Paramètres

string_var

[in][out] La chaîne, qui sera complétée.

add_substring

[in] La chaîne, qui sera ajoutée à la fin de la ligne initiale.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false.Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Exemple:

```
void OnStart()
  {
   long length=1000000;
   string a="a",b="b",c;
//--- premier moyen
   uint start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);

//--- deuxième moyen
   start=GetTickCount();
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   stop=GetTickCount();
   Print("time for 'StringAdd(a,b)' = ",(stop-start)," milliseconds, i = ",i);

//--- troisième moyen
   start=GetTickCount();
   a="a"; // initialisons de nouveau la variable a
   for(i=0;i<length;i++)
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       StringConcatenate(c,a,b);
      }
   stop=GetTickCount();
   Print("time for 'StringConcatenate(c,a,b)' = ",(stop-start)," milliseconds, i = ",i
  }
```

Voir aussi

StringConcatenate

© 2000-2015, MetaQuotes Software Corp.

StringBufferLen

Rend la grandeur du buffer distribué pour la chaîne.

```
int  StringBufferLen(
   string  string_var      // chaîne
   )
```

Paramètres

string_var

[in] La chaîne.

La valeur rendue

La valeur 0 signifie que la chaîne est constante et c'est interdit le contenu du buffer. -1 signifie que la chaîne appartient au terminal de client et le changement du contenu du buffer peut amener aux résultats incertains.

Note

La grandeur du buffer minimal est égal à 16.

Exemple:

```
void OnStart()
  {
   long length=1000;
   string a="a",b="b";
//---
   long i;
   Print("before: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   Print("after: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
  }
```

Voir aussi

StringAdd, StringInit, StringLen, StringFill

© 2000-2015, MetaQuotes Software Corp.

StringCompare

Compare deux chaînes et retourne le résultat de la comparaison comme un entier.

```
int  StringCompare(
   const string&  string1,                // une première chaîne à comparer
   const string&  string2,                // une deuxième chaîne à comparer
   bool           case_sensitive=true     // le mode du compte du registre des lettre
   );
```

Paramètres

string1

[in] Une première chaîne.

string2

[in] Une deuxième chaîne.

case_sensitive=true

[in] Le mode du compte du registre des lettres. Si la valeur est true, c'est "A">"a". Si la valeur est false, c'est "A"="a". Par défaut la valeur du paramètre est true.

La valeur rendue

- -1 (moins un), si string1<string2
- 0 (le zéro), si string1=string2
- 1 (un), si string1>string2

Note

Les chaînes sont comparées caractère par caractère, les caractères sont comparés en ordre alphabétique avec la page de code actuelle.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
void OnStart()
  {
//--- qu'est-ce que est plus grande, la pomme ou la maison?
   string s1="Apple";
   string s2="home";

//--- comparerons en tenant compte du registre
   int result1=StringCompare(s1,s2);
   if(result1>0) PrintFormat("la comparaison en tenant compte du registre: %s > %s",s1
   else
     {
      if(result1<0)PrintFormat("La comparaison en tenant compte du registre: %s < %s",
      else PrintFormat("La comparaison en tenant compte du registre: %s = %s",s1,s2);
     }

//--- comparerons sans tenant compte du registre
   int result2=StringCompare(s1,s2,false);
   if(result2>0) PrintFormat("La comparaison sans tenant compte du registre: %s > %s",
   else
     {
      if(result2<0)PrintFormat("La comparaison sans tenant compte du registre: %s < %s
      else PrintFormat("La comparaison sans tenant compte du registre: %s = %s",s1,s2)
     }
/* Résultat:
    La comparaison en tenant compte du registre: Apple < home
    La comparaison sans tenant compte du registre: Apple < home
*/
  }
```

Voir aussi

Type string, CharToString(), ShortToString(), StringToCharArray(), StringToShortArray(), StringGetCharacter(), Utilisation de la page de code

© 2000-2015, MetaQuotes Software Corp.

StringConcatenate

Forme la chaîne des paramètres transmis et rend la grandeur de la chaîne formée. Les paramètres peuvent avoir n'importe quel type. Le nombre de paramètres ne peut pas être moins 2 et ne peut pas excéder 64.

```
int  StringConcatenate(
   string&  string_var,    // chaîne pour la formation
   void argument1          // premier paramètre de n'importe quel type simple
   void argument2          // deuxième paramètre de n'importe quel type simple
   ...                     // paramètre suivant de n'importe quel type simple
   );
```

Paramètres

string_var

[out] La chaîne qui sera formée à la suite de la concaténation.

argumentN

[in] N'importe quelles valeurs divisées par les virgules.De 2 à 63 paramètres de n'importe quel type simple.

La valeur rendue

Rend la longueur de chaîne, formée par la concaténation de paramètres transformés dans le type string. Les paramètres seront transformés en chaînes selon les mêmes règles que dans les fonctions Print() et Comment().

Voir aussi

StringAdd

© 2000-2015, MetaQuotes Software Corp.

StringFill

Remplit la chaîne indiquée selon la place par les symboles indiqués.

```
bool  StringFill(
   string&   string_var,       // chaîne pour le remplissage
   ushort    character         // symbole, par lequel on remplit la chaîne
   );
```

Paramètres

string_var

[in][out] La chaîne, qui sera remplie par le symbole indiqué.

character

[in] Le symbole, par qui on remplira la chaîne.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false. Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Note

Le remplissage de la chaîne selon la place signifie que les symboles sont insérés directement dans la chaîne sans opérations intermédiaires de la création de la nouvelle chaîne et le copiage. Cela permet de diminuer le temps de travail avec la chaîne dans cette fonction.

Exemple:

```
void OnStart()
  {
   string str;
   StringInit(str,20,'_');
   Print("str = ",str);
   StringFill(str,0);
   Print("str = ",str,": StringBufferLen(str) = ", StringBufferLen(str));
  }
// Résultat
//   str = ____________________
//   str =  : StringBufferLen(str) = 20
//
```

Voir aussi

StringBufferLen, StringLen, StringInit

© 2000-2015, MetaQuotes Software Corp.

StringFind

La recherche de la sous-chaîne dans la chaîne.

```
int  StringFind(
   string  string_value,        // chaîne, où on fait la recherche
   string  match_substring,     // ce qui est recherché
   int     start_pos=0          // par quelle position commencer la recherche
   );
```

Paramètres

string_value

[in] La chaîne, dans laquelle la recherche est faite.

match_substring

[in] La sous-chaîne recherchée.

start_pos=0

[in] La position dans la chaîne, à partir de laquelle on doit commencer la recherche.

La valeur rendue

Rend le numéro de la position dans la chaîne, par laquelle la sous-chaîne recherchée commence, ou -1, si la sous-chaîne n'est pas trouvée.

© 2000-2015, MetaQuotes Software Corp.

StringGetCharacter

Rend la copie de la chaîne avec la valeur changée du symbole dans la position indiquée.

```
ushort  StringGetCharacter(
   string  string_value,     // chaîne
   int     pos               // position du symbole dans la chaîne
   );
```

Paramètres

string_value

[in] La chaîne.

pos

[in] La position du symbole dans la chaîne. Peut être de 0 jusqu'à StringLen(text) -1.

La valeur rendue

Le code du symbole ou 0 en cas d'une erreur. Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

StringInit

Initialise la chaîne par les symboles indiqués et assure la grandeur indiquée de la chaîne.

```
bool  StringInit(
   string&    string_var,       // chaîne pour l'initialisation
   int        new_len=0,        // longueur demandée de la chaîne après l'initialisatio
   ushort     character=0       // symbole par lequel la chaîne sera rempli
   );
```

Paramètres

string_var

[in][out] La chaîne, qui doit être initialisée ou déinitialisé.

new_len=0

[in] La longueur de la chaîne après l'initialisation. Si la longueur est=0, elle déinitialise la chaîne, c'est-à-dire, le buffer de la chaîne est dégagé et l'adresse du buffer se met au zéro.

character=0

[in] Le symbole pour le remplissage de la chaîne.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false. Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Note

Si character=0 et la grandeur new_len>0, le buffer de la chaîne de la longueur indiquée sera distribué et sera rempli par les zéros. La longueur de la chaîne sera égale au zéro, parce que tout le buffer sera rempli par les terminateurs de la chaîne.

Exemple:

```
void OnStart()
  {
//---
   string str;
   StringInit(str,200,0);
   Print("str = ",str,": StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/*  Résultat
str = : StringBufferLen(str) = 200   StringLen(str) = 0
*/
```

Voir aussi

StringBufferLen, StringLen

© 2000-2015, MetaQuotes Software Corp.

StringLen

Rend le nombre de symboles dans la chaîne.

```
int  StringLen(
   string  string_value      // chaîne
   );
```

Paramètres

string_value

[in] La chaîne pour le calcul de la longueur.

La valeur rendue

Le nombre de symboles dans la chaîne sans le zéro terminant.

© 2000-2015, MetaQuotes Software Corp.

StringReplace

Remplace tous les sous-chaînes recherchés dans la chaîne pour la séquence spécifiée des caractères.

```
int  StringReplace(
   string&         str,              // la chaîne, où sera faite le remplacement
   const string    find,             // la sous-chaîne recherchée
   const string    replacement       // la sous-chaîne, qui sera insérée dans les plac
   );
```

Paramètres

str

[in][out] La chaîne dans laquelle il faut faire le changement.

find

[in] La sous-chaîne recherchée pour le remplacement.

replacement

[in] La chaîne qui doit être insérée à la place de la chaîne trouvée.

La valeur rendue

Le nombre de remplacements produits en cas du succès, en cas de l'erreur c'est -1. Pour recevoir le codede l'erreur il faut appeler la fonction GetLastError().

Note

Si la fonction a fonctionné avec succès, mais les remplacements ne sont pas produits (la sous-chaîne remplacée n'est pas trouvée), 0 revient

La cause de l'erreur peuvent être des paramètres incorrects *str* ou *find* (la chaîne vide ou n'est pas initialisée, à voir StringInit()). En outre l'erreur apparaîtra, s'il ne suffit pas la mémoire pour l'achèvement des remplacements.

Exemple:

```
  string text="The quick brown fox jumped over the lazy dog.";
  int replaced=StringReplace(text,"quick","slow");
  replaced+=StringReplace(text,"brown","black");
  replaced+=StringReplace(text,"fox","bear");
  Print("Replaced: ", replaced,". Result=",text);

//  Le résultat
//  Replaced: 3. Result=The slow black bear jumped over the lazy dog.
//
```

Voir aussi

StringSetCharacter(), StringSubstr()

© 2000-2015, MetaQuotes Software Corp.

StringSetCharacter

Rend la copie de la chaîne avec la valeur changée du symbole dans la position indiquée.

```
bool  StringSetCharacter(
   string&   string_var,       // chaîne
   int       pos,              // position
   ushort    character         // symbole
   );
```

Paramètres

string_var

[in][out] La chaîne.

pos

[in] La position du symbole dans la chaîne. Peut être de 0 jusqu'à StringLen(text).

character

[in] Le code de symbole Unicode.

Note

Si la valeur pos est moins que la longueur de la chaîne et la valeur du code de symbole est= 0, la chaîne est coupée (mais la grandeur du buffer, distribuée pour la chaîne reste invariable). La longueur de la chaîne devient égale à la signification pos.

Si la signification du paramètre *pos* est égale à la signification de la longueur de la chaîne, le symbole indiqué est ajouté à la fin de la chaîne,et ainsi la longueur de la chaîne augmente sur une unité.

Exemple:

```
void OnStart()
  {
   string str="0123456789";
   Print("before: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- ajoutez la valeur zéro au milieu de la chaîne
   StringSetCharacter(str,6,0);
   Print("  after: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- ajoutons le symbole à la fin de la chaîne
   int size=StringLen(str);
   StringSetCharacter(str,size,'+');
   Print("addition: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/* Résultat
    before: str = 0123456789 ,StringBufferLen(str) = 0   StringLen(str) = 10
     after: str = 012345 ,StringBufferLen(str) = 16   StringLen(str) = 6
  addition: str = 012345+ ,StringBufferLen(str) = 16   StringLen(str) = 7
```

© 2000-2015, MetaQuotes Software Corp.

```
*/
```

Voir aussi

StringBufferLen, StringLen, StringFill, StringInit

© 2000-2015, MetaQuotes Software Corp.

StringSplit

Reçoit les sous-chaînes selon un séparateur spécifié de la chaîne indiquée et rend le nombre de sous-chaînes reçus.

```
int  StringSplit(
   const string   string_value,       // la chaîne pour la recherche des sous-chaînes
   const ushort   separator,          // le séparateur, à l'aide duquel les sous-chaîn
   string         & result[]          // le tableau transmis selon le lien pour la réc
   );
```

Paramètres

string_value

[in] La chaîne d'où il faut recevoir les sous-chaînes. La chaîne elle-même ne change pas.

pos

[in] Le code du symbole du séparateur. Pour recevoir le code, vous pouvez utiliser la fonction StringGetCharacter().

result[]

[out] Le tableau des chaînes, où se placent les sous-chaînes reçus.

La valeur rendue

Le nombre de chaînes réçues dans le tableau result[]. Si le séparateur dans la chaîne transmise n'est pas trouvée, alors seulement une chaîne initiale sera placée au tableau.

Si la chaîne string_value est vide ou NULL, la fonction rendra le zéro. Dans le cas de l'erreur, la fonction rendra -1. Pour recevoir le codede l'erreur il faut appeler la fonction GetLastError().

Exemple:

```
string to_split="_maman_a_lavé_la_fenêtre_"; // la chaîne pour se diviser en sous-chaî
   string sep="_";                           // le séparateur comme le symbole
   ushort u_sep;                             // le code du symbole du séparateur
   string result[];                          // le tableau pour recevoir les chaînes
   //--- recevrons le code du séparateur
   u_sep=StringGetCharacter(sep,0);
   //--- divisons la chaîne en sous-chaînes
   int k=StringSplit(to_split,u_sep,result);
   //--- déduisons le commentaire
   PrintFormat("On a reçu le nombre de chaînes: %d. On a utilisé le séparateur '%s' av
   //--- maintenant nous déduisons toutes les chaînes reçues
   if(k>0)
     {
      for(int i=0;i<k;i++)
        {
         PrintFormat("result[%d]=\"%s\"",i,result[i]);
        }
     }
```

Voir aussi

StringReplace(), StringSubstr(), StringConcatenate()

© 2000-2015, MetaQuotes Software Corp.

StringSubstr

Extrait une sous-chaîne d'une chaîne de texte à partir de la position spécifiée..

```
string  StringSubstr(
   string  string_value,      // chaîne
   int     start_pos,         // par quelle position commencer
   int     length=-1          // longueur de la chaîne extraite
   );
```

Paramètres

string_value

[in] La chaîne, d'où doit être extrait la sous-chaîne.

start_pos

[in] La position initiale de la sous-chaîne. Peut - être de 0 jusqu'à StringLen(text) -1.

length=-1

[in] La longueur de la sous-chaîne extraite. Si la valeur du paramètre est égale au -1 ou le paramètre n'est pas spécifié, la sous-chaîne sera extrait, à partir de la position indiquée et jusqu'à la fin de la chaîne.

La valeur rendue

La copie du sous-chaîne extrait s'il est possible, autrement revient la chaîne vide.

© 2000-2015, MetaQuotes Software Corp.

StringToLower

Transforme tous les symboles de la chaîne indiquée aux symboles minuscules selon la place.

```
bool  StringToLower(
   string&  string_var      // chaîne à traiter
   );
```

Paramètres

string_var

[in][out] La chaîne.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false.Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

StringToUpper

Transforme tous les symboles de la chaîne indiquée aux symboles majuscules selon la place.

```
bool  StringToUpper(
   string&  string_var      // chaîne à traiter
   );
```

Paramètres

string_var

[in][out] La chaîne.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false.Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

StringTrimLeft

Supprime les symboles de la conversion de chariot, les espaces et les symboles de la tabulation du début de la chaîne jusqu'au premier symbole significatif. La chaîne est modifiée selon la place

```
int  StringTrimLeft(
   string&  string_var      // chaîne à couper
   );
```

Paramètres

string_var

[in][out] La chaîne, qui sera coupé à partir de la gauche.

La valeur rendue

Rend le nombre de symboles coupés.

© 2000-2015, MetaQuotes Software Corp.

StringTrimRight

Supprime les symboles de la conversion de chariot, les espaces et les symboles de la tabulation à la fin de la chaîne. La chaîne est modifiée selon la place.

```
int  StringTrimRight(
   string&  string_var      // chaîne à couper
   );
```

Paramètres

string_var

[in][out] La chaîne, qui sera coupé à droite.

La valeur rendue

Rend le nombre de symboles coupés.

© 2000-2015, MetaQuotes Software Corp.

La date et le temps

Le groupe des fonctions assurant le travail avec les données du type datetime (un nombre entier qui représente le nombre de secondes, passées dès 0 heures du 1 janvier 1970).

Pour organiser les compteurs et la minuterie à haute résolution il faut utiliser la fonction GetTickCount () qui donne la valeur en millisecondes.

Fonction	Action
TimeCurrent	Rend le dernier temps connu du serveur (le temps de l'arrivée de la dernière cotation) dans le format datetime
TimeTradeServer	Rend le temps calculé courant du serveur commercial
TimeLocal	Rend le temps local informatique dans le format datetime
TimeGMT	Rend le temps GMT au format datetime en tenant compte du passage sur l'heure d'hiver ou d'été par le temps local de l'ordinateur, sur lequel le terminal de client a été lancé
TimeDaylightSavings	Rend le signe du passage sur l'heure d'été/ l'hiver
TimeGMTOffset	Rend la différence actuelle entre le temps GMT et le temps local de l'ordinateur en secondes en tenant compte du passage sur l'heure d'hiver ou d'été
TimeToStruct	Produit la conversion de la valeur du type datetime à la variable du type de la structure MqlDateTime
StructToTime	Produit la conversion de la valeur du type de la structure MqlDateTime à la variable du type datetime

© 2000-2015, MetaQuotes Software Corp.

TimeCurrent

Rend le dernier temps connu du serveur, le temps de l'arrivée de la dernière cotation pour un des symboles choisis dans "L'aperçu du marché". Au gestionnaire OnTick() cette fonction rendra le temps du tick traité venant. Dans les autres cas (par exemple, l'appel dans les gestionnaires OnInit(), OnDeinit(), OnTimer() etc) c'est - le temps de l'arrivée de la dernière cotation selon n'importe quel symbole, accessible dans la fenêtre "L'aperçu du marché", ce temps, qui est montré dans le titre de cette fenêtre. La valeur du temps est formée sur un serveur commercial et ne dépend pas des des ajustements du temps sur l'ordinateur de l'utilisateur. Il y a 2 variantes de la fonction.

L'appel sans paramètres

```
datetime  TimeCurrent();
```

L'appel avec le paramètre du type MqlDateTime

```
datetime  TimeCurrent(
   MqlDateTime&  dt_struct      // variable du type de la structure
   );
```

Paramètres

dt_struct

[out] La variable du type de la structure MqlDateTime.

La valeur rendue

La valeur du type datetime

Note

Si la variable du type de la structure MqlDateTime a été transmise comme paramètre, elle s'est remplie de façon appropriée.

Pour organiser les compteurs et la minuterie à haute résolution il faut utiliser la fonction GetTickCount () qui donne la valeur en millisecondes.

При работе в тестере стратегий время последней котировки TimeCurrent() моделируется в соответствии с историческими данными.

© 2000-2015, MetaQuotes Software Corp.

TimeTradeServer

Rend le temps calculé courant du serveur commercial. A la différence de la fonction TimeCurrent(), le calcul de la valeur du temps est produit dans le terminal de client et dépend des réglages du temps de l'ordinateur de l'utilisateur. Il y a 2 variantes de la fonction.

L'appel sans paramètres

```
datetime  TimeTradeServer();
```

L'appel avec le paramètre du type MqlDateTime

```
datetime  TimeTradeServer(
   MqlDateTime&  dt_struct      // variable du type de la structure
   );
```

Paramètres

dt_struct

[out] La variable du type de la structure MqlDateTime.

La valeur rendue

La valeur du type datetime

Note

Si la variable du type de la structure MqlDateTime a été transmise comme paramètre, elle s'est remplie de façon appropriée.

Pour organiser les compteurs et la minuterie à haute résolution il faut utiliser la fonction GetTickCount () qui donne la valeur en millisecondes.

При работе в тестере стратегий TimeTradeServer() всегда равно моделируемому серверному времени TimeCurrent().

© 2000-2015, MetaQuotes Software Corp.

TimeLocal

Rend le temps local de l'ordinateur, sur lequel on a lancé le terminal de client. Il y a 2 variantes de la fonction.

L'appel sans paramètres

```
datetime  TimeLocal();
```

L'appel avec le paramètre du type MqlDateTime

```
datetime  TimeLocal(
   MqlDateTime&  dt_struct      // variable du type de la structure
   );
```

Paramètres

dt_struct

[out] La variable du type de la structure MqlDateTime.

La valeur rendue

La valeur du type datetime

Note

Si la variable du type de la structure MqlDateTime a été transmise comme paramètre, elle s'est remplie de façon appropriée.

Pour organiser les compteurs et la minuterie à haute résolution il faut utiliser la fonction GetTickCount () qui donne la valeur en millisecondes.

При работе в тестере стратегий локальное время TimeLocal() всегда равно моделируемому серверному времени TimeCurrent().

© 2000-2015, MetaQuotes Software Corp.

TimeGMT

Rend le temps GMT au format datetime en tenant compte du passage sur l'heure d'hiver ou d'été par le temps local de l'ordinateur, sur lequel le terminal de client a été lancé. Il y a 2 variantes de la fonction.

L'appel sans paramètres

```
datetime  TimeGMT();
```

L'appel avec le paramètre du type MqlDateTime

```
datetime  TimeGMT(
   MqlDateTime&  dt_struct      // variable du type de la structure
   );
```

Paramètres

dt_struct

[out] La variable du type de la structure MqlDateTime.

La valeur rendue

La valeur du type datetime

Note

Si la variable du type de la structure MqlDateTime a été transmise comme paramètre, elle s'est remplie de façon appropriée.

Pour organiser les compteurs et la minuterie à haute résolution il faut utiliser la fonction GetTickCount () qui donne la valeur en millisecondes.

При работе в тестере стратегий время TimeGMT() всегда равно моделируемому серверному времени TimeTradeServer().

© 2000-2015, MetaQuotes Software Corp.

TimeDaylightSavings

Rend la correction sur l'heure d'été en des secondes, si on a fait le passage à l'heure d'été. Dépend des réglages du temps de l'ordinateur de l'utilisateur.

```
int  TimeDaylightSavings();
```

La valeur rendue

Si on a fait le passage à l'heure d'hiver (standard), revient 0.

© 2000-2015, MetaQuotes Software Corp.

TimeGMTOffset

Rend la différence actuelle entre le temps GMT et le temps local de l'ordinateur en secondes en tenant compte du passage sur l'heure d'hiver ou d'été. Dépend des réglages du temps de l'ordinateur de l'utilisateur.

```
int  TimeGMTOffset();
```

La valeur rendue

La valeur du type int, qui présente la différence courante entre le temps local de l'ordinateur et le temps GMT en secondes.

© 2000-2015, MetaQuotes Software Corp.

TimeToStruct

Produit la conversion de la valeur du type datetime (le nombre de secondes depuis le 01.01.1970) à la variable du type de la structure MqlDateTime.

```
void  TimeToStruct(
   datetime      dt,            // date et temps
   MqlDateTime&  dt_struct      // structure pour l'adoption de valeurs
   );
```

Paramètres

dt

[in] La valeur de la date pour la conversion.

dt_struct

[out] La variable du type de la structure MqlDateTime.

La valeur rendue

Il n'y a pas de valeur rendue.

© 2000-2015, MetaQuotes Software Corp.

StructToTime

Produit la conversion de la valeur du type de la structure MqlDateTime à la variable du type datetime et rend la valeur reçue.

```
datetime  StructToTime(
   MqlDateTime&  dt_struct      // structure de la date et du temps
   );
```

Paramètres

dt_struct

[in] La variable du type de la structure MqlDateTime.

La valeur rendue

La valeur du type datetime, qui contient le nombre de secondes depuis le 01.01.1970.

© 2000-2015, MetaQuotes Software Corp.

L'information sur le compte

Les fonctions qui rendent des paramètres du compte courant.

Fonction	Action
AccountInfoDouble	Rend la valeur du type double de la propriété correspondante du compte
AccountInfoInteger	Rend la valeur du type entier (bool,int ou long) de la propriété correspondante du compte
AccountInfoString	Rend la valeur du type string de la propriété correspondante du compte

© 2000-2015, MetaQuotes Software Corp.

AccountInfoDouble

Rend la valeur de la propriété correspondante du compte.

```
double  AccountInfoDouble(
   int  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. La valeur peut être une des valeurs ENUM_ACCOUNT_INFO_DOUBLE.

La valeur rendue

La valeur du type double.

Exemple:

```
void OnStart()
  {
//--- montrons toute l'information disponible de la fonction AccountInfoDouble()
   printf("ACCOUNT_BALANCE =  %G",AccountInfoDouble(ACCOUNT_BALANCE));
   printf("ACCOUNT_CREDIT =  %G",AccountInfoDouble(ACCOUNT_CREDIT));
   printf("ACCOUNT_PROFIT =  %G",AccountInfoDouble(ACCOUNT_PROFIT));
   printf("ACCOUNT_EQUITY =  %G",AccountInfoDouble(ACCOUNT_EQUITY));
   printf("ACCOUNT_MARGIN =  %G",AccountInfoDouble(ACCOUNT_MARGIN));
   printf("ACCOUNT_FREEMARGIN =  %G",AccountInfoDouble(ACCOUNT_FREEMARGIN));
   printf("ACCOUNT_MARGIN_LEVEL =  %G",AccountInfoDouble(ACCOUNT_MARGIN_LEVEL));
   printf("ACCOUNT_MARGIN_SO_CALL = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL));
   printf("ACCOUNT_MARGIN_SO_SO = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_SO));
  }
```

Voir aussi

SymbolInfoDouble, SymbolInfoString, SymbolInfoInteger, PrintFormat

© 2000-2015, MetaQuotes Software Corp.

AccountInfoInteger

Rend la valeur de la propriété correspondante du compte.

```
long  AccountInfoInteger(
   int  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. La valeur peut être une des valeurs ENUM_ACCOUNT_INFO_INTEGER.

La valeur rendue

La valeur du type long.

Note

La propriété doit être un des types bool, int ou long.

Exemple:

```
void OnStart()
  {
//--- montrons toute l'information accessible de la fonction AccountInfoInteger()
   printf("ACCOUNT_LOGIN =  %d",AccountInfoInteger(ACCOUNT_LOGIN));
   printf("ACCOUNT_LEVERAGE =  %d",AccountInfoInteger(ACCOUNT_LEVERAGE));
   bool thisAccountTradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_ALLOWED);
   bool EATradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_EXPERT);
   ENUM_ACCOUNT_TRADE_MODE tradeMode=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(ACCOU
   ENUM_ACCOUNT_STOPOUT_MODE stopOutMode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoIntege

//--- informons de la possibilité d'exécuter les opérations commerciales
   if(thisAccountTradeAllowed)
      Print("Le commerce pour ce compte est permis");
   else
      Print("Le commerce pour ce compte est interdit!");

//---  éclaircissons - s'il est possible de faire du commerce sur ce compte par les ex
   if(EATradeAllowed)
      Print("Le commerce par les conseillers pour ce compte est permis");
   else
      Print("Le commerce par les conseillers pour ce compte est interdit!");

//--- découvrons le type de compte
   switch(tradeMode)
     {
      case(ACCOUNT_TRADE_MODE_DEMO):
         Print("C'est un compte de démonstration");
         break;
```

© 2000-2015, MetaQuotes Software Corp.

```
      case(ACCOUNT_TRADE_MODE_CONTEST):
         Print("C'est un compte de compétition");
         break;
      default:Print("C'est un compte réel!");
     }

//--- déterminons la mode de la spécification du niveau StopOut
   switch(stopOutMode)
     {
      case(ACCOUNT_STOPOUT_MODE_PERCENT):
         Print("Le niveau StopOut est spécifiée en pourcentage");
         break;
      default:Print("Le niveau StopOut est spécifiée en moyens monétaires");
     }
  }
```

Voir aussi

Information sur le compte

© 2000-2015, MetaQuotes Software Corp.

AccountInfoString

Rend la valeur de la propriété correspondante du compte.

```
string  AccountInfoString(
   int  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. La valeur peut être une des valeurs ENUM_ACCOUNT_INFO_STRING.

La valeur rendue

La valeur du type string.

Exemple:

```
void OnStart()
  {
//--- montrons toute l'information accessible de la fonction AccountInfoString()
   Print("Nom du courtier = ",AccountInfoString(ACCOUNT_COMPANY));
   Print("Devise du dépôt = ",AccountInfoString(ACCOUNT_CURRENCY));
   Print("Nom du client = ",AccountInfoString(ACCOUNT_NAME));
   Print("Nom de serveur commercial = ",AccountInfoString(ACCOUNT_SERVER));
  }
```

Voir aussi

Information sur le compte

© 2000-2015, MetaQuotes Software Corp.

La vérification de l'état

Les fonctions qui rendent des paramètres de l'état courant du terminal de client

Fonction	Action
GetLastError	Rend la valeur de la dernière erreur
IsStopped	Rend true, si on a donné l'ordre de terminer l'exécution du programme mql5
UninitializeReason	Rend le code de la raison de la déinitialisation
TerminalInfoInteger	Rend la valeur du type entier de la propriété correspondante de l'environnement du programme mql5
TerminalInfoDouble	Rend la valeur du type double de la propriété correspondante de l'environnement du programme mql5
TerminalInfoString	Rend la valeur du type string de la propriété correspondante de l'environnement du programme mql5
MQLInfoInteger	Rend la valeur du type entier de la propriété correspondante du programme mql5 lancé
MQLInfoString	Rend la valeur du type string de la propriété correspondante du programme mql5 lancé
Symbol	Rend le nom du symbole du graphique courant
Period	Rend la valeur du temps trame du graphique courant
Digits	Rend le nombre de signes décimaux après la virgule, définissant l'exactitude de la mesure du prix du symbole du graphique courant
Point	Rend la dimension de point de l'instrument courant en devise de la cotation

© 2000-2015, MetaQuotes Software Corp.

GetLastError

Rend le contenu de la variable du système _LastError.

```
int  GetLastError();
```

La valeur rendue

Rend la valeur de la dernière erreur, qui s'est produite pendant l'exécution du programme mql5.

Note

Après l'appel de la fonction, le contenu de la variable _LastError ne se remet pas au zéro. Pour remettre au zéro cette variable il faut appeler cette fonction ResetLastError()

Voir aussi

Codes du retour du serveur commercial

© 2000-2015, MetaQuotes Software Corp.

IsStopped

Vérifie la fermeture forcée du travail du programme mql5.

```
bool  IsStopped();
```

La valeur rendue

Rend true, si la variable du système _StopFlag contient la valeur, qui se diffère du 0. La valeur non zéro s'inscrit à la variable _StopFlag, si on a reçu l'ordre de terminer l'exécution du programme mql5. Dans ce cas il est nécessaire de terminer le travail du programme le plus vite possible, au contraire le programme sera terminé forcément du dehors dans 3 secondes.

© 2000-2015, MetaQuotes Software Corp.

UninitializeReason

Rend le code de la raison de la déinitialisation.

```
int  UninitializeReason();
```

La valeur rendue

Rend la valeur de la variable _UninitReason, qui est formé avant l'appel de la fonction OnDeinit(). La valeur dépend de la raison qui a amené à la déinitialisation.

© 2000-2015, MetaQuotes Software Corp.

TerminalInfoInteger

Rend la valeur d'une propriété correspondante de l'environnement de programme mql5.

```
int  TerminalInfoInteger(
   int  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. Peut être une des valeurs de l'énumération ENUM_TERMINAL_INFO_INTEGER.

La valeur rendue

La valeur du type int.

© 2000-2015, MetaQuotes Software Corp.

TerminalInfoDouble

Возвращает значение соответствующего свойства окружения mql5-программы.

```
double  TerminalInfoDouble(
   int  property_id      // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства. Может быть одним из значений перечисления ENUM_TERMINAL_INFO_DOUBLE.

Возвращаемое значение

Значение типа double.

© 2000-2015, MetaQuotes Software Corp.

TerminalInfoString

La fonction rend la valeur de la propriété correspondante de l'environnement de programme mql5. La propriété doit être du type string

```
string  TerminalInfoString(
   int  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. Peut être une des valeurs de l'énumération ENUM_TERMINAL_INFO_STRING.

La valeur rendue

La valeur du type string.

© 2000-2015, MetaQuotes Software Corp.

MQLInfoInteger

Rend la valeur de la propriété correspondante du programme MQL5 lancé.

```
int  MQLInfoInteger(
   int  property_id      //  identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. Peut être une des valeurs de l'énumération ENUM_MQL_INFO_INTEGER.

La valeur rendue

La valeur du type int.

© 2000-2015, MetaQuotes Software Corp.

MQLInfoString

Rend la valeur de la propriété correspondante du programme MQL5 lancé.

```
string  MQLInfoString(
   int  property_id      // identificateur de la propriété
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété. Peut être une des valeurs de l'énumération ENUM_MQL_INFO_STRING.

La valeur rendue

La valeur du type string.

© 2000-2015, MetaQuotes Software Corp.

Symbol

Rend le nom du symbole courant.

```
string  Symbol();
```

La valeur rendue

La valeur de la variable du système _Symbol, dans laquelle se trouve le nom du sumbole du graphique courant.

© 2000-2015, MetaQuotes Software Corp.

Period

Rend la valeur du temps trame du graphique courant.

```
ENUM_TIMEFRAMES  Period();
```

La valeur rendue

Le contenu de la variable _Period, où se trouve la valeur du temps trame du graphique courant. La valeur peut être une des valeurs de l'énumération ENUM_TIMEFRAMES.

Voir aussi

PeriodSeconds, Les périodes des graphiques, Date et temps, La visibilité des objets

© 2000-2015, MetaQuotes Software Corp.

Digits

Rend le nombre de signes décimaux après la virgule, définissant l'exactitude de la mesure du prix du symbole du graphique courant.

```
int  Digits();
```

La valeur rendue

La valeur de la variable _Digits, qui conserve le nombre de chiffres décimaux après la virgule, déterminant l'exactitude de la mesure du prix du du symbole de graphique actuel.

© 2000-2015, MetaQuotes Software Corp.

Point

Rend la dimension de point de l"instrument courant en devise de la cotation.

```
double  Point();
```

La valeur rendue

La valeur de la variable _Point, où se trouve la grandeur du point de l"instrument courant en devise de la cotation.

© 2000-2015, MetaQuotes Software Corp.

La réception de l'information de marché

Les fonctions pour la réception de l'information sur l'état du marché.

Fonction	Action
SymbolsTotal	Rend le nombre accessible (choisi dans MarketWatch ou tous)des symboles
SymbolName	Rend le nom du symbole indiqué
SymbolSelect	Choisit le symbole dans la fenêtre MarketWatch ou enlève le symbole de la fenêtre
SymbolIsSynchronized	Vérifie le synchronismedes données selon le symbole indiqué dans le terminal avec les données sur le serveur commercial
SymbolInfoDouble	Rend la valeur du type double du symbole indiqué pour la propriété correspondante
SymbolInfoInteger	Rend la valeur du type entier (long, datetime, int ou bool) du symbole indiqué pour la propriété correspondante
SymbolInfoString	Rend la valeur du type string du symbole indiqué pour la propriété correspondante
SymbolInfoMarginRate	Возвращает коэффициенты взимания маржи в зависимости от типа и направления ордера
SymbolInfoTick	Rend les valeurs courantes pour le symbole indiqué dans la variable du type MqlTick
SymbolInfoSessionQuote	Permet de recevoir le temps du début et le temps de la fin de la session de cotation indiquée pour le symbole et le jour de la semaine indiqués.
SymbolInfoSessionTrade	Permet de recevoir le temps du début et le temps de la fin de la session commerciale indiquée pour le symbole et le jour de la semaine indiqués.
MarketBookAdd	Assure l'ouverture du profondeur de marché selon l'instrument indiqué, ainsi que produit la souscription sur la réception des avis du changement du profondeur de marché indiqué
MarketBookRelease	Assure la clôture du profondeur de marché selon l'instrument, ainsi qui élimine la souscription sur la réception des avis du changement du profondeur de marché indiqué
MarketBookGet	Rend le tableau des structures du type MqlBookInfo, contenant les enregistrements du

© 2000-2015, MetaQuotes Software Corp.

	profondeur de marché du symbole indiqué

© 2000-2015, MetaQuotes Software Corp.

SymbolsTotal

Rend le nombre accessible (choisi dans MarketWatch ou tous)des symboles.

```
int  SymbolsTotal(
   bool  selected      // true - seulement les symboles dans MarketWatch
   );
```

Paramètres

selected

[in] Le mode de la demande. Peut accepter les valeurs true ou false.

La valeur rendue

Si le paramètre selected est égal à true, revient le nombre de symboles choisis dans MarketWatch. Si la valeur false, revient le total de tous les symboles.

© 2000-2015, MetaQuotes Software Corp.

SymbolName

Rend le nom du symbole indiqué.

```
string  SymbolName(
   int   pos,          // numéro dans la liste
   bool  selected      // true - seulement les symboles dans MarketWatch
   );
```

Paramètres

pos

[in] Le numéro du symbole par ordre.

selected

[in] Le mode de la demande. Si la valeur est true, on prend le symbole de la liste des choisis dans MarketWatch. Si la valeur est false, on prend le symbole de la liste commune.

La valeur rendue

La valeur du type string avec le nom du symbole.

© 2000-2015, MetaQuotes Software Corp.

SymbolSelect

Choisit le symbole dans la fenêtre MarketWatch ou enlève le symbole de la fenêtre.

```
bool  SymbolSelect(
   string  name,        // nom du symbole
   bool    select       // brancher ou débrancher
   );
```

Paramètres

name

[in] Le nom du symbole.

select

[in] Le commutateur. Si la valeur est false, le symbole doit être enlevé de la fenêtre MarketWatch, dans le cas contraire le symbole doit être choisi à la fenêtre MarketWatch. Le symbole ne peut pas être enlevé, s'il y a des graphiques ouverts avec ce symbole ou il y a des positions ouvertes dans ce symbole.

La valeur rendue

En cas de l'échec la fonction rend false.

© 2000-2015, MetaQuotes Software Corp.

SymbolIsSynchronized

Contrôle le fait de la synchronisation des données selon le symbole indiqué dans le terminal avec les données sur le serveur commercial

```
bool  SymbolIsSynchronized(
   string  name,        // nom du symbole
   );
```

Paramètres

name

[in] Le nom du symbole.

La valeur rendue

Si les données sont synchronisées, rend true, ou autrement rend false.

Voir aussi

SymbolInfoInteger, L'organisation de l'accès aux données

© 2000-2015, MetaQuotes Software Corp.

SymbolInfoDouble

Rend la propriété correspondante du symbole indiqué. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
double  SymbolInfoDouble(
   string                    name,        // symbole
   ENUM_SYMBOL_INFO_DOUBLE   prop_id      // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  SymbolInfoDouble(
   string                    name,        // symbole
   ENUM_SYMBOL_INFO_DOUBLE   prop_id,     // identificateur de la propriété
   double&                   double_var   // acceptons par ici la valeur de la propriété
   );
```

Paramètres

name

[in] Le nom du symbole.

prop_id

[in] L'identificateur de la propriété du symbole. La valeur peut être une des valeurs ENUM_SYMBOL_INFO_DOUBLE.

double_var

[out] La variable du type double, qui prend la valeur de la propriété demandée.

La valeur rendue

La valeur du type double. En cas d'une mauvaise exécution on peut recevoir l'information sur l'erreurà l'aide de la fonctionGetLastError():

- 5040 - le paramètre de chaîne incorrecte pour l'indication du nom du symbole,
- 4301 - un symbole inconnu (l'instrument financier),
- 4302 - le symbole n'est pas choisi dans "l'Aperçu du marché" (est absent dans la liste des accessibles),
- 4303 - l'identificateur incorrecte de la propriété du symbole.

Note

Si la fonction est utilisée pour la réception de l'information sur un dernier tick, il vaut mieux utiliserSymbolInfoTick(). Il est possible que selon ce symbole dès le moment de la connexion du terminal au compte commercial il n'y avait pas encore aucune cotation. Dans un tel cas la valeur demandée sera incertaine.

Il suffit d'utiliser dans la plupart des cas la fonction SymbolInfoTick(), qui permet de recevoir pour un appel les valeurs Ask, Bid, Last, Volume et le temps de l'arrivée du dernier tick.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
void OnTick()
  {
//--- recevrons le spread des propriétés du symbole
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d des points\r\n",
                            spreadfloat?"flottant":"fixé",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- maintenant calculerons nous-même le spread
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"spread calculé = "+(string)spread_points+" des points";
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

SymbolInfoInteger

Rend la propriété correspondante du symbole indiqué. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
long  SymbolInfoInteger(
   string                     name,      // symbole
   ENUM_SYMBOL_INFO_INTEGER   prop_id    // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  SymbolInfoInteger(
   string                     name,      // symbole
   ENUM_SYMBOL_INFO_INTEGER   prop_id,   // identificateur de la propriété
   long&                      long_var   // acceptons par ici la valeur de la propriété
   );
```

Paramètres

name

[in] Le nom du symbole.

prop_id

[in] L'identificateur de la propriété du symbole. La valeur peut être une des valeurs ENUM_SYMBOL_INFO_INTEGER.

long_var

[out] La valeur du type long, recevant la valeur de la propriété demandée.

La valeur rendue

La valeur du type long. En cas d'une mauvaise exécution on peut recevoir l'information sur l'erreurà l'aide de la fonctionGetLastError():

- 5040 - le paramètre de chaîne incorrecte pour l'indication du nom du symbole,
- 4301 - un symbole inconnu (l'instrument financier),
- 4302 - le symbole n'est pas choisi dans "l'Aperçu du marché" (est absent dans la liste des accessibles),
- 4303 - l'identificateur incorrecte de la propriété du symbole.

Note

Si la fonction est utilisée pour la réception de l'information sur un dernier tick, il vaut mieux utiliserSymbolInfoTick(). Il est possible que selon ce symbole dès le moment de la connexion du terminal au compte commercial il n'y avait pas encore aucune cotation. Dans un tel cas la valeur demandée sera incertaine.

Il suffit d'utiliser dans la plupart des cas la fonction SymbolInfoTick(), qui permet de recevoir pour un appel les valeurs Ask, Bid, Last, Volume et le temps de l'arrivée du dernier tick.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
void OnTick()
  {
//--- recevrons le spread des propriétés du symbole
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d des points\r\n",
                            spreadfloat?"flottant":"fixé",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- maintenant calculerons nous-même le spread
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"spread calculé = "+(string)spread_points+" des points";
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

SymbolInfoString

Rend la propriété correspondante du symbole indiqué. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
string  SymbolInfoString(
   string                    name,       // symbole
   ENUM_SYMBOL_INFO_STRING   prop_id     // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  SymbolInfoString(
   string                    name,       // symbole
   ENUM_SYMBOL_INFO_STRING   prop_id,    // identificateur de la propriété
   string&                   string_var  // acceptons par ici la valeur de la propriét
   );
```

Paramètres

name

[in] Le nom du symbole.

prop_id

[in] L'identificateur de la propriété du symbole. La valeur peut être une des valeurs ENUM_SYMBOL_INFO_STRING.

string_var

[out] La valeur du type string, recevant la valeur de la propriété demandée.

La valeur rendue

La valeur du type string. En cas d'une mauvaise exécution on peut recevoir l'information sur l'erreurà l'aide de la fonctionGetLastError():

- 5040 - le paramètre de chaîne incorrecte pour l'indication du nom du symbole,
- 4301 - un symbole inconnu (l'instrument financier),
- 4302 - le symbole n'est pas choisi dans "l'Aperçu du marché" (est absent dans la liste des accessibles),
- 4303 - l'identificateur incorrecte de la propriété du symbole.

Note

Si la fonction est utilisée pour la réception de l'information sur un dernier tick, il vaut mieux utiliserSymbolInfoTick(). Il est possible que selon ce symbole dès le moment de la connexion du terminal au compte commercial il n'y avait pas encore aucune cotation. Dans un tel cas la valeur demandée sera incertaine.

Il suffit d'utiliser dans la plupart des cas la fonction SymbolInfoTick(), qui permet de recevoir pour un appel les valeurs Ask, Bid, Last, Volume et le temps de l'arrivée du dernier tick.

© 2000-2015, MetaQuotes Software Corp.

SymbolInfoMarginRate

Возвращает коэффициенты взимания маржи в зависимости от типа и направления ордера.

```
bool  SymbolInfoMarginRate(
   string             name,                    // символ
   ENUM_ORDER_TYPE    order_type,              // тип ордера
   double&            initial_margin_rate,     // коэффициент взимания начальной марж
   double&            maintenance_margin_rate  // коэффициент взимания поддерживающей
   );
```

Параметры

name

[in] Имя символа.

order_type

[in] Тип ордера.

initial_margin_rate

[in] Переменная типа double для получения коэффициента взимания начальной маржи. Начальная маржа - это размер гарантийной суммы под совершение сделки объемом в 1 лот соответствующего направления. Умножая коэффициент на начальную маржу, мы можем получить размер средств, который будет зарезервирован на счете при размещении ордера указанного типа.

maintenance_margin_rate

[out] Переменная типа double для получения коэффициента взимания поддерживающей маржи. Поддерживающая маржа - это размер минимальной суммы для поддержания открытой позиции объемом в 1 лот соответствующего направления. Умножая коэффициент на поддерживающую маржу, мы можем получить размер средств, который будет зарезервирован на счете после срабатывания ордера указанного типа.

Возвращаемое значение

Возвращает true в случае удачного выполнения запроса свойств, иначе false.

© 2000-2015, MetaQuotes Software Corp.

SymbolInfoTick

Rend les prix courants pour le symbole indiqué dans la variable du type MqlTick.

```
bool  SymbolInfoTick(
   string    symbol,      // symbole
   MqlTick&  tick         // référence à la structure
   );
```

Paramètres

symbol

[in] Le nom du symbole.

tick

[out] Le lien à la structure du type MqlTick, où seront placés les prix courants et le temps de la dernière mise à jour des prix.

La valeur rendue

Rend true en cas du succès, autrement - false.

© 2000-2015, MetaQuotes Software Corp.

SymbolInfoSessionQuote

Permet de recevoir le temps du début et le temps de la fin de la session de cotation indiquée pour le symbole et le jour de la semaine indiqués.

```
bool  SymbolInfoSessionQuote(
   string            name,               // le nom du symbole
   ENUM_DAY_OF_WEEK  day_of_week,        // le jour de la semaine
   uint              session_index,      // le numéro de la session
   datetime&         from,               // le temps du début de la session
   datetime&         to                  // le temps de la fin de la session
   );
```

Les paramètres

name

[in] Le nom du symbole.

ENUM_DAY_OF_WEEK

[in] Le jour de la semaine, la valeur de l'énumération ENUM_DAY_OF_WEEK.

uint

[in] Le numéro de série de la session, pour laquelle il est nécessaire de recevoir le temps du début et le temps de la fin. L'indexation des sessions commence par 0.

from

[out] Le temps du début de la session de 00 heures 00 minutes, il faut ignorer la date dans la valeur reçue.

to

[out] Le temps de la fin de la session de 00 heures 00 minutes, il faut ignorer la date dans la valeur reçue.

La valeur rendue

Si les données sont reçues pour les sessions indiquées, le symbole et le jour de la semaine, on rend true, autrement rend false.

Voir aussi

L'information sur l'instrument, TimeToStruct, La structure de la date

© 2000-2015, MetaQuotes Software Corp.

SymbolInfoSessionTrade

Permet de recevoir le temps du début et le temps de la fin de la session commerciale indiquée pour le symbole et le jour de la semaine indiqués.

```
bool  SymbolInfoSessionTrade(
   string            name,               // le nom du symbole
   ENUM_DAY_OF_WEEK  day_of_week,        // le jour de la semaine
   uint              session_index,      // le numéro de la session
   datetime&         from,               // le temps du début de la session
   datetime&         to                  // le temps de la fin de la session
   );
```

Les paramètres

name

[in] Le nom du symbole.

ENUM_DAY_OF_WEEK

[in] Le jour de la semaine, la valeur de l'énumération ENUM_DAY_OF_WEEK.

uint

[in] Le numéro de série de la session, pour laquelle il est nécessaire de recevoir le temps du début et le temps de la fin. L'indexation des sessions commence par 0.

from

[out] Le temps du début de la session de 00 heures 00 minutes, il faut ignorer la date dans la valeur reçue.

to

[out] Le temps de la fin de la session de 00 heures 00 minutes, il faut ignorer la date dans la valeur reçue.

La valeur rendue

Si les données sont reçues pour les sessions indiquées, le symbole et le jour de la semaine, on rend true, autrement rend false.

Voir aussi

L'information sur l'instrument, TimeToStruct, La structure de la date

© 2000-2015, MetaQuotes Software Corp.

MarketBookAdd

Assure l'ouverture du profondeur de marché selon l'instrument indiqué, ainsi que produit la souscription sur la réception des avis du changement du profondeur de marché indiqué.

```
bool  MarketBookAdd(
   string  symbol      // symbole
   );
```

Paramètres

symbol

[in] Le nom du symbole, dont le profondeur de marché est supposé d'utiliser dans l'expert donné ou le script.

La valeur rendue

La valeur true en cas de l'ouverture fructueuse, autrement false.

Note

Normalement, on doit appeler cette fonction de la fonction OnInit() ou dans le constructeur de la classe. Pour le traitement des avis venant dans le programme de l'expert doit être la fonction void OnBookEvent(string& symbol).

Voir aussi

La structure du profondeur de marché, Les structures et les classes

© 2000-2015, MetaQuotes Software Corp.

MarketBookRelease

Assure la clôture du profondeur de marché selon l'instrument indiqué, ainsi qu'élimine la souscription sur la réception des avis du changement du profondeur de marché indiqué.

```
bool  MarketBookRelease(
   string  symbol      // nom du symbole
   );
```

Paramètres

symbol

[in] Le nom du symbole.

La valeur rendue

La valeur true en cas de la clôture fructueuse, autrement false.

Note

Normalement, on doit appeler cette fonction de la fonction OnDeinit() dans le cas où dans la fonction OnInit() a été appelée la fonction MarketBookAdd(). Ou elle doit appeler du destructeur de la classe, si dans le constructeur de cette classe s'appelle la fonction correspondante MarketBookAdd().

Voir aussi

La structure du profondeur de marché, Les structures et les classes

© 2000-2015, MetaQuotes Software Corp.

MarketBookGet

Rend le tableau des structures du type MqlBookInfo, contenant les enregistrements du profondeur de marché du symbole indiqué.

```
bool  MarketBookGet(
   string         symbol,      // symbol
   MqlBookInfo&   book[]       // référence au tableau
   );
```

Paramètres

symbol

[in] Le nom du symbole.

book[]

[out] La référence au tableau des enregistrements du profondeur de marché. Le tableau peut être d'avance distribué pour le nombre suffisant des enregistrements. Si le tableau dynamique n'était pas d'avance distribué dans la mémoire d'exploitation, le terminal de client distribuera ce tableau lui-même.

La valeur rendue

Rend true en cas du succès, autrement - false.

Note

Le profondeur de marché doit être préalablement ouvert par la fonction MarketBookAdd().

Exemple:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo jusqu'au ",Symbol());
      for(int i=0;i<size;i++)
        {
         Print(i+": ",priceArray[i].price
               +"    Volume = "+priceArray[i].volume,
               " type = ",priceArray[i].type);
        }
     }
   else
     {
      Print("On n'a pas réussi à recevoir le contenu du profondeur de marché selon le
     }
```

Voir aussi

La structure du profondeur de marché, Les structures et les classes

© 2000-2015, MetaQuotes Software Corp.

L'accès aux séries temporelles et les données des indicateurs

Les fonctions pour le travail avec les séries temporelles et les indicateurs. La série temporelle se distingue du tableau ordinaire par ce que l'indexation des éléments de la série temporelle est produite de la fin du tableau vers le début (des données les plus récentes aux les plus vieux). Pour le copiage des valeurs des séries temporelles et des indicateurs il est recommandé d'utiliser seulement les tableaux dynamiques, puisque les fonctions du copiage distribuent indépendamment la grandeur nécessaire des tableaux-récepteurs des valeurs.

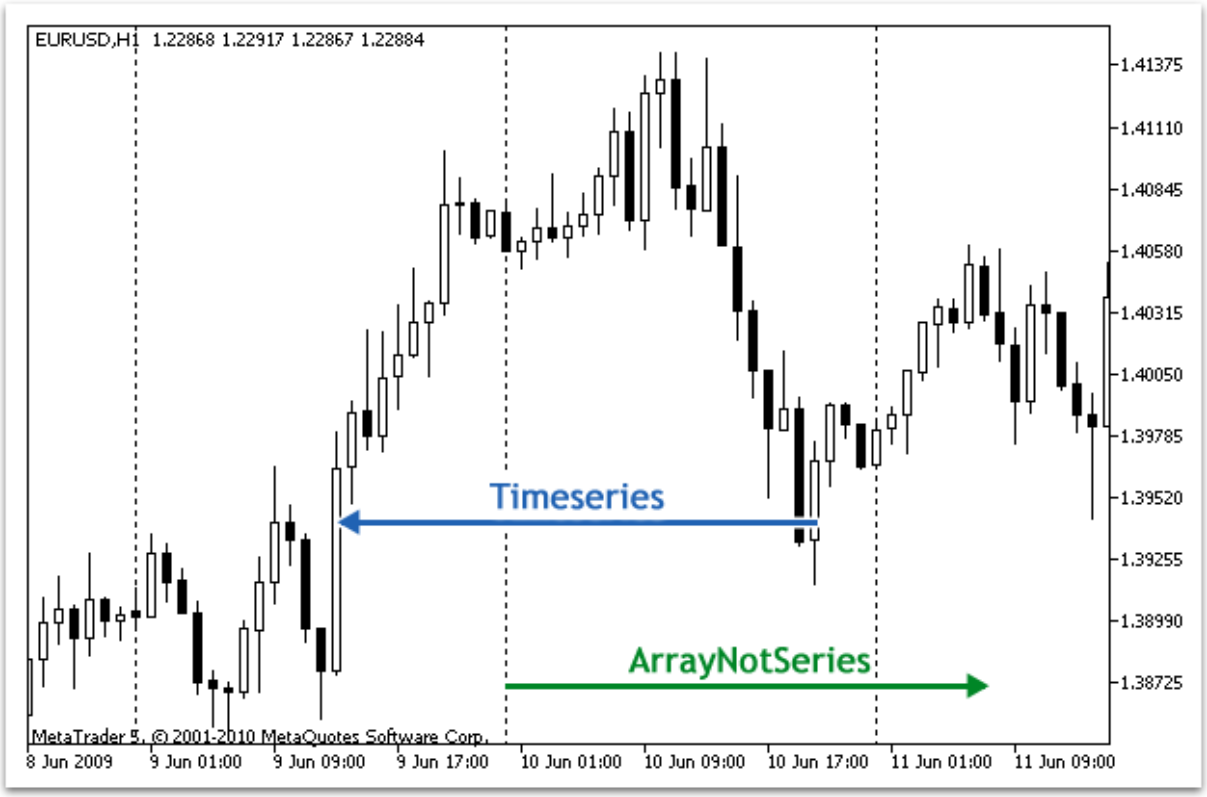
Il y a **une importante exception** de cette règle: s'il est nécessaire de faire le copiage des séries temporelles et les valeurs des indicateurs souvent, par exemple, à chaque appel de OnTick() dans les experts ou à chaque appel de OnCalculate() les indicateurs, dans ce cas il vaut mieux d'utiliser les tableaux statiquement distribués, parce que **les opérations de la distribution de la mémoire** sous les tableaux dynamiques **demandent le temps supplémentaire** et cela se manifesta au test et à l'optimisation des experts.

A l'utilisation des fonctions de l'accès aux séries temporelles et les valeurs des indicateurs il est nécessaire de prendre en considération la direction de l'indexation, c'est en détail décrit dans le paragraphe La direction de l'indexation dans les tableaux et dans les séries temporelles.

L'accès aux données des indicateurs et des séries temporelles se réalise indépendamment du fait de la disponibilité des données demandées (un soi-disant l'accès asynchrone). C'est critiquement important pour le calcul des indicateurs d'utilisateur, c'est pourquoi en l'absence des données demandées de la fonction comme *Copy...()* rend l'erreur tout de suite.Cependant à l'accès des experts et des scripts on produit quelques tentatives de la réception des données avec une petite pause, appelé à assurer le temps nécessaire au chargement des séries temporelles manquantes ou au calcul des valeurs des indicateurs.

Le paragraphe l'Organisation de l'accès aux données décrit des détails de la réception, de la conservation et de la demande des données de prix dans le terminal de client MetaTrader 5.

© 2000-2015, MetaQuotes Software Corp.



C'est formé historiquement ainsi que l'accès aux données dans le tableau de prix était produit de la fin des données. Physiquement, les nouvelles données sont toujours écrites à la fin de tableau, mais l'index du tableau est toujours égal au zéro. L'index 0 dans le tableau-la série temporelle signifie les données de la barre courante, c'est-à-dire la barre, qui correspond à l'intervalle inachevée du temps dans ce temps trame.

Le temps trame - la période du temps, pendant laquelle on forme une barre de prix; il est prédéterminé au total les 21 temps trames standard.

Fonction	Action
SeriesInfoInteger	Rend l'information sur l'état des données historiques
Barres	Rend le nombre de barres aux histoires selon le symbole correspondant et la période
BarsCalculated	Rend la quantité de données calculées dans le tampon d'indicateur ou-1 en cas de l'erreur (les données ne sont pas encore calculées)
IndicatorCreate	Rend le handle de l'indicateur indiqué technique créé à la base du tableau des paramètres du type MqlParam
IndicatorParameters	Rend le nombre de paramètres d'entrée de l'indicateur selon le handle indiqué, ainsi que les valeurs elles-mêmes et le type des paramètres
IndicatorRelease	Supprime le handle de l'indicateur et libère la

© 2000-2015, MetaQuotes Software Corp.

	partie calculée de l'indicateur, si personne ne se sert plus d'elle
CopyBuffer	Reçoit au tableau les données du tampon indiqué de l'indicateur indiqué
CopyRates	Reçoit au tableau les données historiques de la structure Rates pour le symbole et la période indiqués
CopyTime	Reçoit au tableau les données historiques par le temps de l'ouverture des barres selon le symbole et la période correspondants
CopyOpen	Reçoit au tableau les données historiques au prix de l'ouverture des barres selon le symbole et la période correspondants
CopyHigh	Reçoit au tableau les données historiques selon le prix maximum des barres selon le symbole et la période correspondants
CopyLow	Reçoit au tableau les données historiques selon le prix minimum des barres selon le symbole et la période correspondants
CopyClose	Reçoit au tableau les données historiques au prix de la clôture des barres selon le symbole et la période correspondants
CopyTickVolume	Reçoit au tableau les données historiques par les volumes des ticks pour le symbole et la période correspondants
CopyRealVolume	Reçoit au tableau les données historiques par les volumes commerciaux pour le symbole et la période correspondants
CopySpread	Reçoit au tableau les données historiques selon les spread pour le symbole et la période correspondants
CopyTicks	Получает в массив тики, накопленные терминалом за текущую рабочую сессию

Malgré le fait que par la fonction ArraySetAsSeries() on puisse spécifier aux tableaux le moyen de l'accès aux éléments comme pour la série temporelle, il faut se rappeler que physiquement les éléments du tableau se trouvent toujours en même ordre, change seulement la direction de l'indexation. Pour la démonstration de ce fait on peut accomplir l'exemple:

```
   datetime TimeAsSeries[];
//--- établissons l'accès au tableau comme à la série temporelle
   ArraySetAsSeries(TimeAsSeries,true);
   ResetLastError();
   int copied=CopyTime(NULL,0,0,10,TimeAsSeries);
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(copied<=0)
     {
      Print("On n'a pas réussi à copier le temps de l'ouverture pour les dernières 10
      return;
     }
   Print("TimeCurrent = ",TimeCurrent());
   Print("ArraySize(Time) = ",ArraySize(TimeAsSeries));
   int size=ArraySize(TimeAsSeries);
   for(int i=0;i<size;i++)
     {
      Print("TimeAsSeries["+i+"] = ",TimeAsSeries[i]);
     }

   datetime ArrayNotSeries[];
   ArraySetAsSeries(ArrayNotSeries,false);
   ResetLastError();
   copied=CopyTime(NULL,0,0,10,ArrayNotSeries);
   if(copied<=0)
     {
      Print("On n'a pas réussi à copier le temps de l'ouverture pour les dernières 10
      return;
     }
   size=ArraySize(ArrayNotSeries);
   for(int i=size-1;i>=0;i--)
     {
      Print("ArrayNotSeries["+i+"] = ",ArrayNotSeries[i]);
     }
```

On produit finalement la conclusion semblable à cela:

```
TimeCurrent = 2009.06.11 14:16:23
ArraySize(Time) = 10
TimeAsSeries[0] = 2009.06.11 14:00:00
TimeAsSeries[1] = 2009.06.11 13:00:00
TimeAsSeries[2] = 2009.06.11 12:00:00
TimeAsSeries[3] = 2009.06.11 11:00:00
TimeAsSeries[4] = 2009.06.11 10:00:00
TimeAsSeries[5] = 2009.06.11 09:00:00
TimeAsSeries[6] = 2009.06.11 08:00:00
TimeAsSeries[7] = 2009.06.11 07:00:00
TimeAsSeries[8] = 2009.06.11 06:00:00
TimeAsSeries[9] = 2009.06.11 05:00:00

ArrayNotSeries[9] = 2009.06.11 14:00:00
ArrayNotSeries[8] = 2009.06.11 13:00:00
ArrayNotSeries[7] = 2009.06.11 12:00:00
ArrayNotSeries[6] = 2009.06.11 11:00:00
ArrayNotSeries[5] = 2009.06.11 10:00:00
ArrayNotSeries[4] = 2009.06.11 09:00:00
```

© 2000-2015, MetaQuotes Software Corp.

```
ArrayNotSeries[3] = 2009.06.11 08:00:00
ArrayNotSeries[2] = 2009.06.11 07:00:00
ArrayNotSeries[1] = 2009.06.11 06:00:00
ArrayNotSeries[0] = 2009.06.11 05:00:00
```

Comme on voit des résultats de la conclusion, pour le tableau TimeAsSeries avec la croissance de l'index diminue la valeur du temps se trouvant sous cet index, c'est-à-dire nous nous avançons du présent vers le passé. Pour le tableau ordinaire ArrayNotSeries tout au contraire - avec la croissance de l'index nous avançons du passé vers le présent.

Voir aussi

ArrayIsDynamic, ArrayGetAsSeries, ArraySetAsSeries, ArrayIsSeries

© 2000-2015, MetaQuotes Software Corp.

La direction de l'indexation dans les tableaux, les tampons et les séries temporelles

Tous les tableaux et les tampons d'indicateur ont la direction de l'indexation de gauche à droite par défaut. L'index du premier élément est toujours égal au zéro. Ainsi, le premier élément du tableau ou du tampon d'indicateur avec l'index 0 se trouve par défaut à l'extrême gauche de la position, le dernier élément se trouve à l'extrême droite de la position.

Le tampon d'indicateur présente de lui-même le tableau dynamique du type double, dont la grandeur est dirigé par le terminal de client pour qu'il corresponde toujours au nombre de barres, sur lesquelles l'indicateur est calculé. Le tableau ordinaire dynamique du type double est assigné à titre du tampon d'indicateur à l'aide de la fonction SetIndexBuffer(). Pour les tampons d'indicateur il ne faut pas spécifier la grandeur à l'aide de la fonction ArrayResize(), le système exécutant du terminal s'en occupera.

Les séries temporelles se présentent les tableaux avec l'indexation inverse, c'est-à-dire le premier élément de la série temporelle se trouve sur l'extrême droite de la position, mais le dernier élément de la série temporelle se trouve sur l'extrême gauche de la position. Puisque les séries temporelles sont destinées à la conservation des données historiques de prix selon les instruments financiers et contiennent absolument l'information sur le temps, on peut dire que les données les plus nouvelles dans la série temporelle se trouvent dans la position droite extrême, mais les vieilles positions les plus dans l'extrême gauche.

C'est pourquoi, l'élément avec l'index le zéro dans la série temporelle contient l'information sur la dernière cotation selon l'instrument. Si la série temporelle présente les renseignements selon le temps trame de jour, sur la position zéro il y a les données du jour courant inachevé, mais sur la position avec l'index un se trouvent les données de la journée d'hier.

Le changement de la direction de l'indexation

La fonction ArraySetAsSeries() permet de changer le moyen de l'accès aux éléments du tableau dynamique, mais physiquement l'ordre de la conservation des données dans la mémoire de l'ordinateur ne change pas. Cette fonction change simplement le moyen de l'adressage vers les éléments de tableau, c'est pourquoi au copiage d'un tableau à l'autre à l'aide de la fonction ArrayCopy() le contenu du tableau-récepteur ne dépendra pas de la direction de l'indexation dans le tableau-source.

On ne peut pas changer la direction de l'indexation pour les tableaux statiquement distribués. Même si le tableau était transmis à titre du paramètre à la fonction, et à l'intérieur de cette fonction les tentatives de changer la direction de l'indexation n'amèneront à rien.

Pour les tampons d'indicateur, comme pour les tampons ordinaires, il est permis d'établir aussi la direction de l'indexation à l'envers comme dans la série temporelle, c'est-à-dire, l'appel à la position zéro dans le tampon d'indicateur signifiera dans ce cas l'appel à la dernière valeur dans le tampon correspondant d'indicateur et cela correspondra à la valeur de l'indicateur sur une dernière barre. En même temps physiquement le placement de données dans le tampon d'indicateur restera invariable, comme c'était déjà mentionné.

La réception des données de prix dans les indicateurs

Dans chaque indicateur d'utilisateur doit absolument assister la fonction OnCalculate(), à laquelle on transmet les données de prix nécessaires au calcul des valeur dans les tampons d'indicateur. On peut

© 2000-2015, MetaQuotes Software Corp.

savoir la direction de l'indexation dans ces tableaux transmis à l'aide de la fonction ArrayGetAsSeries().

Les tableaux transmis à la fonction reflètent les données de prix, c'est-à-dire ces tableaux ont l'index de la série temporelle et la fonction ArrayIsSeries() rendra true à la vérification de ces tableaux. Mais néanmoins, il est nécessaire dans tous les cas de contrôler la direction de l'indexation seulement la fonction ArrayGetAsSeries().

Pour ne pas dépendre de défauts, il est nécessaire absolument d'appeler la fonction ArraySetAsSeries() pour les tableaux, avec lesquels on suppose à travailler et établir la direction demandée de l'indexation.

La réception des données de prix et les valeurs des indicateurs

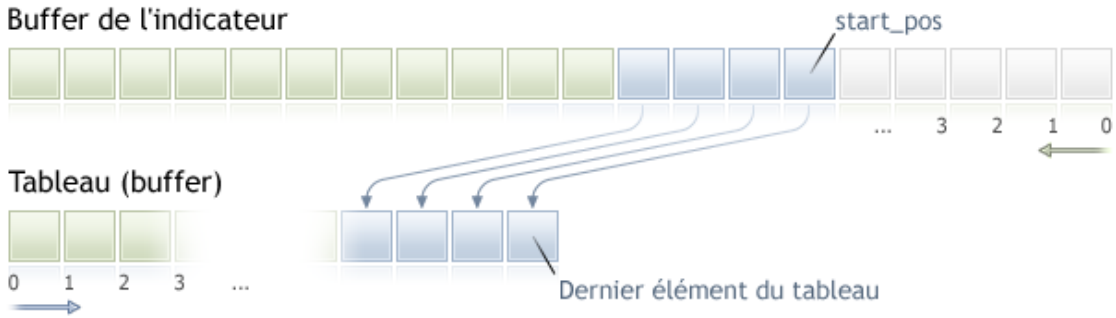
Dans les experts, les indicateurs et les scripts tous les tableaux ont la direction de l'indexation de gauche à droite par défaut. En cas de nécessité dans n'importe quel programme mql5 on peut demander les valeurs des séries temporelles selon n'importe quel caractère et le timeframe, ainsi que les valeurs des indicateurs calculés sur n'importe quel caractère et le temps trame.

Pour la réception de telles données servent les fonctions Copy...():

- CopyBuffer – le copiage des valeurs du tampon d'indicateur au tableau du type double;
- CopyRates – le copiage de l'histoire de prix au tableau des structures MqlRates;
- CopyTime – le copiage des valeurs Time au tableau du type datetime;
- CopyOpen – le copiage des valeurs Open au tableau du type double;
- CopyHigh – le copiage des valeurs High au tableau du type double;
- CopyLow – le copiage des valeurs Low au tableau du type double;
- CopyClose – le copiage des valeurs Close au tableau du type double;
- CopyTickVolume – le copiage des volumes des ticks du type long;
- CopyRealVolume – le copiage des volumes boursiers au tableau du type long;
- CopySpread – le copiage de l'histoire des spreads au tableau du type int;

Toutes ces fonctions travaillent également, et voilà pourquoi c'est suffisant d'examiner le mécanisme de la réception des données à l'exemple CopyBuffer(). Il est sous-entendu que toutes les données demandées ont la direction de l'indexation, comme dans la série temporelle, il est sous-entendu au cela que dans la position avec l'index 0 (zéro) se trouvent les données de la barre courante inachevée. Pour recevoir l'accès à ces données, il faut copier le volume nécessaire des données au tableau-récepteur, par exemple, au tableau *buffer*.

© 2000-2015, MetaQuotes Software Corp.



Au copiage il est nécessaire d'indiquer la position du départ dans le tableau-source, d'où on copiera les données au tableau-récepteur. En cas du succès dans le tableau-destinataire on copie le nombre indiqué d'éléments du tableau-source, dans ce cas du tampon d'indicateur. De plus indépendamment du fait, quelle direction de l'indexation est établie dans le tableau -récepteur, le copiage est produit toujours comme c'est indiqué sur le dessin.

Si les données de prix sont attendues à manipuler dans une boucle avec un grand nombre d'itérations, il est recommandé de vérifier le fait de l'arrêt forcé du programme en utilisant la fonction IsStopped():

```
int copied=CopyBuffer(ma_handle,// le handle de l'indicateur
                      0,        // l'index du tampon d'indicateur
                      0,        // la position de départ pour le copiage
                      number,   // le nombre de valeurs pour le copiage
                      Buffer    // le tableau - destinataire des valeurs
                      );
if(copied<0) return;
int k=0;
while(k<copied && !IsStopped())
  {
   //--- recevrons la valeur pour l'index k
   double value=Buffer[k];
   // ...
   // le travail avec la valeur value
   k++;
  }
```

Exemple:

```
input int per=10; // période de l'exposant
int ma_handle;    // handle de l'indicateur
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   ma_handle=iMA(_Symbol,0,per,0,MODE_EMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---
   double ema[10];
   int copied=CopyBuffer(ma_handle,// handle de l'indicateur
                         0,        // index du tampon d'indicateur
                         0,        // position de départ pour le copiage
                         10,       // nombre de valeurs pour le copiage
                         ema       // tableau-récepteur des valeurs
                         );
   if(copied<0) return;
// .... code ultérieur
  }
```

Voir aussi

L'organisation de l'accès aux données

© 2000-2015, MetaQuotes Software Corp.

L'organisation de l'accès aux données

Dans ce paragraphe on examine les questions liées à la réception, la conservation et les demandes des données de prix (les séries temporelles).

La réception des données du serveur commercial

Avant que les données de prix soient accessibles dans le terminal MetaTrader 5, il est nécessaire de les recevoir et traiter. Pour la réception des données on demande la connexion au serveur commercial MetaTrader 5. Les données entrent du serveur à la demande du terminal en forme des blocs économiquement emballés des barres momentanées.

Le mécanisme de la référence au serveur pour les données ne dépend pas du fait comment comment la demande a été initiée, par l'utilisateur à la navigation selon le graphique ou par le moyen de programme dans le langage MQL5.

La conservation des données intermédiaires

Les données reçues du serveur sont dépaquetées automatiquement et se gardent dans le format spécial intermédiaire HCC. Les données selon chaque symbole s'écrivent au dossier séparé *le répertoire_du terminal*\bases*le nom_du serveur*\history*le nom_du symbole*.Par exemple, les données selon le symbole EURUSD du serveur commercial MetaQuotes-Demo se trouveront dans le dossier *le répertoire_du terminal*\bases\MetaQuotes-Demo\history\EURUSD\.

Les données s'inscrivent aux fichiers avec l'extension .hcc, chaque fichier garde les données des barres momentanées pour un an. Par exemple, le fichier 2009.hcc dans le dossier EURUSD contient les barres momentanées selon le symbole EURUSD pour l'an 2009. Ces fichiers sont utilisés pour la préparation des données de prix selon tous les temps trames et ne sont pas destinés à l'accès direct.

La réception des données du temps trame nécessaire des données intermédiaires

Les fichiers de service au format HCCjouent le rôle de la source des données pour la construction des données de prix selon les temps trames demandés au format HC. Les données au format HC sont les séries temporelles, au maximum préparées pour l'accès rapide. Ils sont créés seulement à la demande du graphique ou du programme mql5 dans le volume n'excédant pas les valeurs du paramètre "Max bars in charts",et se gardent pour l'utilisation ultérieure dans les fichiers avec l'extension hc.

Pour l'économie des ressources les données selon le temps trame sont chargées et se trouvent dans la mémoire RAM seulement par nécessité, en absence de longue durée des appels aux données il y a leur déchargement de la mémoire RAM avec le sauvegarde au fichier. Pour chaque temps trame les données se préparent indépendamment de la présence des données déjà prêtes pour les autres temps trames. Les règles de la formation et l'accessibilité des données sont identiques pour tous les temps trames. C'est-à-dire malgré que l'unité de la conservation des données au format HCC est la barre momentanée, la présence des données au format HCC ne signifie pas la présence et l'accessibilité dans le même volume des données du temps trame M1 au format HC.

La réception des nouvelles données du serveur appelle la mise à jour automatique des données utilisées de prix au format HC selon tous les temps trames et la recalculation de tous les indicateurs, qui les utilisent évidemment à titre des données d'entrée pour le compte.

© 2000-2015, MetaQuotes Software Corp.

Le paramètre "Max bars in chart"

Le paramètre "Max bars in charts" limite le nombre accessible de barres au format HC pour les graphiques, les indicateurs et les programmes mql5. Cette limitations agit pour les données de tous les temps trames, et est destiné tout d'abord, à l'économie des ressources.

En établissant de grandes valeurs du paramètre donné, il faut se rappeler qu'en présence de l'histoire assez profonde des données de prix pour les temps trames cadets la dépense de la mémoire sur la conservation des séries temporelles et des tampons des indicateurs peut être centaines mégabytes et atteindre la limitation de la mémoire RAM pour le programme du terminal de client (2Gb pour les applications MS Windows de 32-bits).

Le changement du paramètre "Max bars in charts"entre en vigueur après le lancement du terminal de client. Même le changement de ce paramètre ne provoque pas ni l'appel automatique au serveur pour les données supplémentaires, ni des formations des barres supplémentaires des séries temporelles. La demande des données supplémentaires de prix près du serveur et la mise à jour des séries temporelles en tenant compte d'une nouvelle limitation qui se passera en cas du défilement du graphique au domaine des données manquants ou en cas de la demande des données manquants du programme mql5.

Le volume des données demandées chez le serveur correspond au nombre demandé de barres du temps trame donné en tenant compte de la valeur du paramètre "Max bars in charts". La limitation donnée par le paramètre, n'est pas rigide, et dans certains cas le nombre de barres accessibles selon le temps trame peut être un peu plus que la valeur courante du paramètre.

L'accessibilité des données

La présence des données au format HCC ou même au format prêt pour l'utilisation HC ne signifie pas toujours l'accessibilité absolue de ces données à l'affichage sur le graphique ou pour l'utilisation dans les programmes mql5.

A l'accès aux données de prix ou aux valeurs des indicateurs des programmes mql5 il faut se rappeler que l'on ne garantit pas leur accessibilité au moment défini du temps, ou du moment défini du temps. C'est raccordé avec le fait qu'avec le but de l'économie des ressources à MetaTrader 5 on ne stocke pas la copie complète des données demandées pour le programme mql5, mais on donne l'accès direct à la base de données du terminal.

L'histoire de prix selon tous les temps trames s'est construite des données communes au format HCC et chaque mise à jour des données du serveur amène à la mise à jour des données selon tous les temps trames et au recalcul des indicateurs. De ce fait l'accès aux données peut être fermé même si ces données étaient disponibles il y a un moment.

La synchronisation des données du terminal et les données du serveur

Puisque le programme mql5 peut s'adresser aux données selon n'importe quel symbole et temps trame, c'est-à-dire il y a la probabilité que les données de la série temporelle demandée ne sont pas encore formées dans le terminal ou les données demandées de prix ne sont pas synchronisées avec le serveur commercial. Dans ce cas-là il est difficile de prédire le temps d'attente de la disponibilité des données.

Les algorithmes avec l'utilisation des boucles de l'attente de la disponibilité des données ne sont pas la meilleure décision. La seule exception dans ce cas —ces sont les scripts, puisqu'ils n'ont pas d'autre choix de l'algorithme en vue de l'absence du traitement des événements. Pour les indicateurs

© 2000-2015, MetaQuotes Software Corp.

d'utilisateur les algorithmes semblables, comme les autres boucles de l'attente, ne sont pas recommandés catégoriquement, puisque ils amènent à l'arrêt du compte de tous les indicateurs et à un autre traitement des données de prix selon le symbole donné.

Pour les experts et les indicateurs d'utilisateur il vaut mieux utiliser le modèle événementiel du traitement. Si au traitement de l'événement OnTick() ou OnCalculate() on ne réussit pas à recevoir toutes les données nécessaires de la série temporelle demandée, en comptant sur l'apparition de l'accès aux données à l'appel suivant du gestionnaire.

L'exemple d'un script pour ajouter l'histoire

Examinerons l'exemple du script, qui accomplit la demande sur la réception de l'histoire du serveur commercial selon l'instrument indiqué. Le script est destiné au lancement de l'instrument demandé sur le graphique, le temps trame n'a pas d'importance, parce que comme il a été mentionné ci-dessus, les données de prix sont reçues d'un serveur commercial en forme des données emballées momentanées, à partir des lesquelles toutes les séries temporelles prédéfinies sont construites après.

Écrivons toutes les actions de la réception des données en forme de la fonction séparée CheckLoadHistory(symbol, timeframe, start_date):

```
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
  }
```

La fonction CheckLoadHistory() est formée comme universel, dont on peut appeler de n'importe quel programme (l'expert, le script ou l'indicateur), et c'est pour cela elle demande trois paramètres d'entrée: le nom du symbole, la période et la date initiale, d'où on a besoin de l'histoire de prix.

insérerons dans le code de la fonction tous les vérifications nécessaires, avant de demander l'histoire manquante. Avant tout il est nécessaire de se persuader que le nom du symbole et la valeur de la période sont correct:

```
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
```

Ensuite nous nous persuaderons que le symbole indiqué est accessible dans la fenêtre MarketWatch, c'est-à-dire, l'histoire selon le symbole donné sera accessible à la demande au serveur commercial. S'il est absent là, ajoutons le symbole à la fenêtre à l'aide de la fonction SymbolSelect().

```
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
```

Maintenant il est nécessaire de recevoir la date initiale selon l'histoire déjà existante pour la paire indiquée le symbole/la période. Il est possible que la valeur du paramètre d'entrée startdate, transmis à la fonction CheckLoadHistory() entre se trouve dans l'intervalle à l'histoire déjà accessible, et alors on ne faudra pas de faire aucune demande au serveur commercial. La fonction SeriesInfoInteger() avec le modificateur SERIES_FIRSTDATE est destinée à la réception de la toute première date selon le caractère-la période pour le moment.

```
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
```

© 2000-2015, MetaQuotes Software Corp.

Une vérification suivante importante — c'est la vérification du type du programme, d'où la fonction est appelée. Rappellerons que l'expédition de la demande pour mettre à jour les séries temporelles avec la même période que celle de l'indicateur, qui appelle la mise à jour, n'est pas souhaitable. Le caractère indésirable de la demande des données selon le même symbole-la période celui de l'indicateur s'explique par le fait que la mise à jour des données historiques est produite dans le même thread, où l'indicateur travaille. C'est pourquoi il y a une grande probabilité de clinch. Pour la vérification nous utilisons la fonction MQL5InfoInteger() avec le modificateur MQL5_PROGRAM_TYPE.

```
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sym
      return(-4);
```

Si toutes les vérifications ont été passées avec succès, faisons la dernière tentative se passer sans s'adresser au serveur commercial. D'abord apprenons la date initiale, pour laquelle les données momentanées au format HCC sont accessibles. Nous demanderons cette valeur par la fonction SeriesInfoInteger() avec le modificateur SERIES_TERMINAL_FIRSTDATE et la comparons de nouveau avec la valeur du paramètre start_date.

```
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- there is loaded data to build timeseries
      if(first_date>0)
        {
         //--- force timeseries build
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- check date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
```

Si après toutes les vérifications le thread de l'exécution se trouve toujours dans le corps de la fonction CheckLoadHistory(), donc, il y a une nécessité de demander les données manquant de prix du serveur commercial.D'abord nous apprenons la valeur "Max bars in chart" à l'aide de la fonction TerminalInfoInteger():

```
  int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
```

Nous en avons besoin pour ne pas demander les données supplémentaires. Trouvons après la première date dans l'histoire selon le symbole sur le serveur commercial (indépendamment de la période) utilisant la fonction déjà connue SeriesInfoInteger() avec le modificateur SERIES_SERVER_FIRSTDATE.

```
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
```

Puisque la requête est l'opération asynchrone, la fonction est appelée dans la boucle avec un petit retard de 5 millisecondes jusqu'à ce que la variable first_server_date ne recevra pas la valeur ou l'exécution de la boucle ne sera pas interrompu par l'utilisateur (IsStopped() dans ce cas rendra la valeur true). Indiquerons la valeur correcte de la date initiale, à partir de laquelle nous demandons du serveur commercial les données de prix.

```
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Warning: first server date ",first_server_date,
            " for ",symbol," does not match to first series date ",first_date);
```

© 2000-2015, MetaQuotes Software Corp.

Si soudain la date initiale *first_server_date* sur le serveur serra moins que la date initiale *first_date* selon le symbole au format HCC, un message approprié sera déduit au journal.

Maintenant nous sommes prêts à faire une demande au serveur commercial pour recevoir les données manquantes de prix. faisons la requête en forme de la boucle et nous commencerons à remplir son corps:

```
   while(!IsStopped())
     {
      //1. attendre la synchronisation entre la série temporelle reconstruite et l'his
      //2. recevoir le nombre courant des barres dans une série temporelle donnée
      //   Si "bars" est plus que "Max_bars_in_chart", on peut sortir, le travail a fi
      //3. Recevrons la date initiale first_date à la série temporelle reconstruite et
      //   si first_date est moins que start_date, on peut sortir, le travail a fini
      //4. Demanderons une nouvelle portion de l'histoire a 100 barres du serveur comm
      //   accessible sous le numéro "bars"
     }
```

Les trois premiers points sont exécutés par les moyens déjà connus.

```
   while(!IsStopped())
     {
      //--- 1.attendre la fin du procès de la reconstruction de la série temporelle
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCRONIZED) && !IsStopped())
         Sleep(5);
      //--- 2.demanderons combien de barres nous avons maintenant
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         //--- il y a plus de barres, qu'on peut afficher sur le graphique, sortons
         if(bars>=max_bars) return(-2);
         //--- 3. apprenons la date courante initiale dans une série temporelle
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            // date initiale est plus précoce, qu'on a demandé, la tâche est accomplie
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //4. Nous demanderons une nouvelle portion de l'histoire à 100 barres
      //   du serveur commercial de la dernière barre accessible sous le numéro "bars"
     }
```

Il y avait un quatrième dernier point — la requête directe de l'histoire. Nous ne pouvons pas directement nous adresser au serveur, mais chaque fonction- Copy au manque de l'histoire au format HCC le terminal initie automatiquement l'envoi d'une telle requête du terminal au serveur commercial. Puisque le temps de la première date initiale dans la variable *first_date* est le critère le plus simple et naturel pour l'évaluation du degré de l'exécution de la requête, c'est sera le plus simple d'utiliser la fonction CopyTime().

En appelant des fonctions qui copient n'importe quelles données de la série temporelle, il faut noter que le paramètre *start* (le numéro de la barre par qui on commence le copiage des données de prix) doit être toujours dans la limite de l'histoire disponible du terminal. Si nous avons seulement 100 barres, il ne faut pas tenter de copier 300 barres, à partir de la barre avec l'index 500. Une telle requête sera perçue comme erroné et ne sera pas traité, c'est-à-dire aucune histoire ne sera chargée d'un serveur commercial.

© 2000-2015, MetaQuotes Software Corp.

C'est pourquoi nous copions les 100 barres, à partir de la barre avec l'index "bars". Cela assurera le chargement fluent de l'histoire du serveur commercial, En fait un peu plus que les 100 barres demandées sera chargé, le serveur rend l'histoire avec le stock.

```
   int copied=CopyTime(symbol,period,bars,100,times);
```

Après l'opération du copiage il est nécessaire d'analyser le nombre d'éléments copiés, si la tentative a été ratée, la valeur de la variable *copied* sera égale au zéro et la valeur du compteur fail_cnt sera augmentée sur 1. Après les 100 essais ratés le travail de la fonction sera interrompu.

```
int fail_cnt=0;
...
   int copied=CopyTime(symbol,period,bars,100,times);
   if(copied>0)
     {
      //--- vérifions les données
      if(times[0]<=start_date)  return(0);  // valeur copiée est moins, est prête
      if(bars+copied>=max_bars) return(-2); //les barres sont devenues plus, qu'est pl
      fail_cnt=0;
     }
   else
     {
      //--- pas plus de 100 essais ratés de suite
      fail_cnt++;
      if(fail_cnt>=100) return(-5);
      Sleep(10);
     }
```

Ainsi, non seulement organisé le traitement correct de la situation actuelle à chaque moment de l'exécution en fonction, mais encore le code de retour revient aussi, que nous pouvons traiter après l'appel de la fonction CheckLoadHistory() pour la réception l'information supplémentaire. Par exemple, ainsi:

```
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Symbole inconnu ",InpLoadedSymbol);
      case -2 : Print("Les barres demandées sont beaucoup plus qu'on peut afficher su
      case -3 : Print("Exécution était interrompue par l'utilisateur");
      case -4 : Print("Indicateur ne doit pas charger les données personnelles");
      case -5 : Print("Chargement s'est terminé par l'échec");
      case  0 : Print("Toutes les données sont chargées");
      case  1 : Print("Il suffit déjà des données qui se trouvent à la série temporell
      case  2 : Print("La série temporelle est construite des données disponibles du t
      default : Print("Résultat de l'exécution n'est pas défini");
     }
```

Le code complet de la fonction est donné dans l'exemple du script, qui présente le moyen juste de l'organisation de l'accès à n'importe quelles données avec le traitement du résultat de la requête.

Code:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                                TestLoadHistory.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.02"
#property script_show_inputs
//--- input parameters
input string          InpLoadedSymbol="NZDUSD";   // Symbol to be load
input ENUM_TIMEFRAMES InpLoadedPeriod=PERIOD_H1;  // Period to be load
input datetime        InpStartDate=D'2006.01.01'; // Start date
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   Print("Start load ",InpLoadedSymbol+", "+GetPeriodName(InpLoadedPeriod)," from ",In
//---
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Unknown symbol ",InpLoadedSymbol);              break;
      case -2 : Print("Requested bars more than max bars in chart"); break;
      case -3 : Print("Program was stopped");                         break;
      case -4 : Print("Indicator shouldn't load its own data");       break;
      case -5 : Print("Load failed");                                 break;
      case  0 : Print("Loaded OK");                                   break;
      case  1 : Print("Loaded previously");                           break;
      case  2 : Print("Loaded previously and built");                 break;
      default : Print("Unknown result");
     }
//---
   datetime first_date;
   SeriesInfoInteger(InpLoadedSymbol,InpLoadedPeriod,SERIES_FIRSTDATE,first_date);
   int bars=Bars(InpLoadedSymbol,InpLoadedPeriod);
   Print("First date",first_date,"-",bars,"bars");
//---
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
   datetime first_date=0;
   datetime times[100];
//--- check symbol & period
   if(symbol==NULL || symbol="") symbol=Symbol();
   if(period==PERIOD_CURRENT)    period=Period();
//--- check if symbol is selected in the MarketWatch
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
//--- check if data is present
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
//--- don't ask for load of its own data if it is an indicator
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sym
```

© 2000-2015, MetaQuotes Software Corp.

```
      return(-4);
//--- second attempt
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- there is loaded data to build timeseries
      if(first_date>0)
        {
         //--- force timeseries build
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- check date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
//--- max bars in chart from terminal options
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
//--- load symbol history info
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
//--- fix start date for loading
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Warning: first server date ",first_server_date,
```

© 2000-2015, MetaQuotes Software Corp.

```
            " for ",symbol," does not match to first series date ",first_date);
//--- load data step by step
   int fail_cnt=0;
   while(!IsStopped())
     {
      //--- wait for timeseries build
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCRONIZED) && !IsStopped())
         Sleep(5);
      //--- ask for built bars
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         if(bars>=max_bars) return(-2);
         //--- ask for first date
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //--- copying of next part forces data loading
      int copied=CopyTime(symbol,period,bars,100,times);
      if(copied>0)
        {
         //--- check for data
         if(times[0]<=start_date)  return(0);
         if(bars+copied>=max_bars) return(-2);
         fail_cnt=0;
        }
      else
        {
         //--- no more than 100 failed attempts
         fail_cnt++;
         if(fail_cnt>=100) return(-5);
         Sleep(10);
        }
     }
//--- stopped
   return(-3);
  }
//+------------------------------------------------------------------+
//| rend la valeur de la chaîne de la période                        |
//+------------------------------------------------------------------+
string GetPeriodName(ENUM_TIMEFRAMES period)
  {
   if(period==PERIOD_CURRENT) period=Period();
//---
   switch(period)
     {
      case PERIOD_M1:  return("M1");
      case PERIOD_M2:  return("M2");
      case PERIOD_M3:  return("M3");
      case PERIOD_M4:  return("M4");
      case PERIOD_M5:  return("M5");
      case PERIOD_M6:  return("M6");
      case PERIOD_M10: return("M10");
      case PERIOD_M12: return("M12");
      case PERIOD_M15: return("M15");
      case PERIOD_M20: return("M20");
      case PERIOD_M30: return("M30");
      case PERIOD_H1:  return("H1");
      case PERIOD_H2:  return("H2");
      case PERIOD_H3:  return("H3");
      case PERIOD_H4:  return("H4");
```

© 2000-2015, MetaQuotes Software Corp.

```
      case PERIOD_H6:  return("H6");
      case PERIOD_H8:  return("H8");
      case PERIOD_H12: return("H12");
      case PERIOD_D1:  return("Daily");
      case PERIOD_W1:  return("Weekly");
      case PERIOD_MN1: return("Monthly");
     }
//---
   return("unknown period");
  }
```

© 2000-2015, MetaQuotes Software Corp.

SeriesInfoInteger

Rend l'information sur l'état des données historiques. Il y a 2 variantes de la fonction.

Rend directement la valeur de la propriété.

```
long  SeriesInfoInteger(
   string                      symbol_name,     // nom du symbole
   ENUM_TIMEFRAMES             temps trame,       // période
   ENUM_SERIES_INFO_INTEGER    prop_id,         // identificateur de la propriété
   );
```

2. Rend true ou false selon du succès de l'exécution de la fonction.

```
bool  SeriesInfoInteger(
   string                      symbol_name,     // nom du symbole
   ENUM_TIMEFRAMES             temps trame,       // période
   ENUM_SERIES_INFO_INTEGER    prop_id,         // identificateur de la propriété
   long&                       long_var         // variable pour la réception de l'info
   );
```

Paramètres

symbol_name

[in] Le symbole.

temps trame

[in] La période.

prop_id

[in] L'identificateur de la propriété demandée, la valeur de l'énumération ENUM_SERIES_INFO_INTEGER.

long_var

[out] La variable, auquelle la valeur de la propriété demandée est placée.

La valeur rendue

La valeur du type long pour la première variante de l'appel.

Pour la deuxième variante de l'appel rend true, si la propriété donnée est supportée et la valeur était placé à la variable long_var, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
void OnStart()
  {
//---
   Print("Le nombre de barres selon le caractère-la période pour le moment = ",
         SeriesInfoInteger(Symbol(),0,SERIES_BARS_COUNT));

   Print("Première date selon le caractère-la période pour le moment = ",
         (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));

   Print("Première date à l'histoire selon le symbole sur le serveur = ",
         (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));

   Print("Données selon le symbole sont synchronisées = ",
         (bool)SeriesInfoInteger(Symbol(),0,SERIES_SYNCRONIZED));
  }
```

© 2000-2015, MetaQuotes Software Corp.

Barres

Rend le nombre de barres dans l'histoire selon le symbole correspondant la période. Il y a 2 variantes de la fonction.

Demander le nombre de tous les barres à l'histoire

```
int  Barres(
   string           symbol_name,     // nom du symbole
   ENUM_TIMEFRAMES  temps trame,       // période
   );
```

Demander le nombre de barres à l'intervalle donné

```
int  Barres(
   string           symbol_name,     // nom du symbole
   ENUM_TIMEFRAMES  temps trame,       // période
   datetime         start_time,      // de quelle date
   datetime         stop_time        // jusqu'à quelle date
   );
```

Paramètres

symbol_name

[in] Le symbole.

temps trame

[in] La période.

start_time

[in] Le temps de la barre correspondante au premier élément.

stop_time

[in] Le temps de la barre correspondante au dernier élément.

La valeur rendue

Si on indique les paramètres start_time et stop_time, la fonction rend le nombre de barres dans le domaine des dates. Si ces paramètres ne sont pas indiqués, la fonction rend le nombre total des barres.

Note

Si les données pour la série temporelle avec les paramètres indiqués à l'appel de la fonction Bars() ne sont pas encore formées dans le terminal, ou les données de la série temporelle au moment de l'appel de la fonction ne sont pas synchronisées_ avec le serveur commercial, la fonction rendra la valeur nulle.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
   int bars=Bars(_Symbol,_Period);
   if(bars>0)
     {
      Print("Le nombre de barres dans l'histoire du terminal selon le caractère-la pé
     }
   else  //il n'y a pas de barres accessibles
     {
      //--- probablement, les données selon le symbole ne sont pas synchronisées avec
      bool synchronized=false;
      //---compteur de la boucle
      int attempts=0;
      // faisons les 5 tentatives d'attendre la synchronisation
      while(attempts<5)
        {
         if(SeriesInfoInteger(Symbol(),0,SERIES_SYNCRONIZED))
           {
            //--- il y a une synchronisation, sortons
            synchronized=true;
            break;
           }
         //--- augmentons le compteur
         attempts++;
         //--- attendrons 10 millisecondes avant l'itération suivante
         Sleep(10);
        }
      //--- nous sommes sortis de la boucle du fait de la synchronisation
      if(synchronized)
        {
         Print("Le nombre de barres dans l'histoire du terminal selon le symbole-la pé
         Print("Date première dans l'histoire du terminal selon le symbole-la période
               (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));
         Print("Date première dans l'histoire selon le symbole au serveur = ",
               (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));
        }
      //--- synchronisation des données n'était pas atteinte
      else
        {
         Print("On n'a pas réussi à recevoir le nombre de bars sur ",_Symbol);
        }
     }
```

Voir aussi

Fonctions de traitement des événements

© 2000-2015, MetaQuotes Software Corp.

BarsCalculated

Rend le nombre de données comptées pour l'indicateur demandé.

```
int  BarsCalculated(
   int       indicator_handle,     // handle de l'indicateur
   );
```

Paramètres

indicator_handle

[in] Le handle de l'indicateur, reçu par la fonction correspondante d'indicateur.

La valeur rendue

Rend le nombre de données comptées dans le tampon d'indicateur ou -1en cas de l'erreur (les données ne sont pas encore comptées).

Note

La fonction est utile dans les cas où il est nécessaire de recevoir les données de l'indicateur à la fois après sa création (la réception du handle de l'indicateur).

Exemple:

```
void OnStart()
  {
   double Ups[];
//--- mettons le critère de la série temporelle pour les tableaux
   ArraySetAsSeries(Ups,true);
//--- créons le handle de l'indicateur Fractals
   int FractalsHandle=iFractals(NULL,0);
//--- réinitialisons le code d'erreur
   ResetLastError();
//--- tenterons de copier les valeurs de l'indicateur
   int i,copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Sleep(50);
      for(i=0;i<100;i++)
        {
         if(BarsCalculated(FractalsHandle)>0)
            break;
         Sleep(50);
        }
      copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
      if(copied<=0)
        {
         Print("On n'a pas réussi à copier les fractals supérieurs. Error = ",GetLastE
         "i = ",i,"    copied = ",copied);
         return;
        }
```

© 2000-2015, MetaQuotes Software Corp.

```
      else
        Print("On a réussi à copier les fractals supérieurs. ",
        "i = ",i,"    copied = ",copied);
    }
  else Print("On a réussi à copier les fractals supérieurs. ArraySize = ",ArraySize(U
}
```

© 2000-2015, MetaQuotes Software Corp.

IndicatorCreate

Rend le handle de l'indicateur indiqué technique créé à la base du tableau des paramètres du type MqlParam.

```
int  IndicatorCreate(
   string           symbol,                    // nom du symbole
   ENUM_TIMEFRAMES  period,                    // période
   ENUM_INDICATOR   indicator_type,            // type de l'indicateur de l"ér
   int              parameters_cnt=0,          // nombre de paramètres
   const MqlParam&  parameters_array[]=NULL,   // tableau des paramètres
   );
```

Paramètres

symbol

[in] L'indicateur sera calculé sur les données du nom symbolique de l'instrument. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeur de l"énumération ENUM_TIMEFRAMES, 0 signifie le temps trame courant.

indicator_type

[in] Le type de l'indicateur peut accepter une des valeur de l"énumération ENUM_INDICATOR.

parameters_cnt

[in] Le nombre de paramètres transmis dans le tableau parameters_array[]. Les éléments de tableau ont le type spécial de la structure MqlParam. Par défaut la valeur nulle - les paramètres ne sont pas transmis. Si on a indiqué le nombre non zéro des paramètres, le paramètre *parameters_array* est obligatoire. On peut transmettre pas plus de 256 paramètres.

parameters_array[]=NULL

[in] Le tableau du type MqlParam, dont les éléments contiennent le type et la valeur de chaque paramètre d'entrée de l'indicateur technique.

La valeur rendue

Rend le handle de l'indicateur indiqué technique,en cas de l'échec rend INVALID_HANDLE.

Note

Si le handle de l'ndicateur du type IND_CUSTOM est crée, le champ *type* du premier élément de tableau des paramètres d'entrée *parameters_array* doit avoir obligatoirement la valeur TYPE_STRING de l'énumération ENUM_DATATYPE, et le champ *string_value* du premier élément doit contenir le nom de l'indicateur d'utilisateur. L'indicateur d'utilisateur doit être compilé (le fichier avec l'extension EX5) et se trouver dans le répertoire MQL5/Indicators du terminal de client ou au sous-répertoire.

Les indicateurs nécessaires au test sont définis automatiquement de l'appel des fonctions, iCustom(), si le paramètre correspondant est spécifié par la ligne constante. Pour les autres cas (l'utilisation de la fonction IndicatorCreate() ou l'utilisation de la ligne non constante dans le paramètre qui spécifie le nom de l'indicateur) la propriété donnée #property tester_indicator est

© 2000-2015, MetaQuotes Software Corp.

nécessaire:

```
#property tester_indicator "indicator_name.ex5"
```

Si dans l'indicateur d'utilisateur on utilise la première forme de l'appel, tà la transmission des paramètres d'entrée par le dernier paramètre on pourra indiquer en supplément sur quelles données il sera calculé. Si le paramètre "Apply to" n'est pas indiqué évidemment, le calcul est produit selon les valeurs PRICE_CLOSE par défaut.

Exemple:

```
void OnStart()
  {
   MqlParam params[];
   int      h_MA,h_MACD;
//--- create iMA("EURUSD",PERIOD_M15,8,0,MODE_EMA,PRICE_CLOSE);
   ArrayResize(params,4);
//--- set ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=8;
//--- set ma_shift
   params[1].type         =TYPE_INT;
   params[1].integer_value=0;
//--- set ma_method
   params[2].type         =TYPE_INT;
   params[2].integer_value=MODE_EMA;
//--- set applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=PRICE_CLOSE;
//--- create MA
   h_MA=IndicatorCreate("EURUSD",PERIOD_M15,IND_MA,4,params);
//--- create iMACD("EURUSD",PERIOD_M15,12,26,9,h_MA);
   ArrayResize(params,4);
//--- set fast ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=12;
//--- set slow ma_period
   params[1].type         =TYPE_INT;
   params[1].integer_value=26;
//--- set smooth period for difference
   params[2].type         =TYPE_INT;
   params[2].integer_value=9;
//--- set indicator handle as applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=h_MA;
//--- create MACD based on moving average
   h_MACD=IndicatorCreate("EURUSD",PERIOD_M15,IND_MACD,4,params);
//--- use indicators
//--- . . .
//--- release indicators (first h_MACD)
   IndicatorRelease(h_MACD);
   IndicatorRelease(h_MA);
  }
```

© 2000-2015, MetaQuotes Software Corp.

IndicatorParameters

Rend le nombre de paramètres d'entrée de l'indicateur selon le handle indiqué, ainsi que les valeurs elles-mêmes et le type des paramètres.

```
int  IndicatorParameters(
   int               indicator_handle,     // le handle de l'indicateur
   ENUM_INDICATOR&   indicator_type,       // la variable pour la réception du type de
   MqlParam&         parameters[]          // le tableau pour la réception des paramèt
   );
```

Paramètres

indicator_handle

[in] Le handle de l'indicateur, dont il est nécessaire de savoir la quantité de paramètres, sur lesquels il a été calculé.

indicator_type

[out] La variable du type ENUM_INDICATOR, où sera enregistré le type de l'indicateur.

parameters[]

[out] Le tableau dynamique pour la réception des valeurs du type MqlParam, à qui la liste des paramètres de l'indicateur sera enregistrée. La fonction IndicatorParameters() rend la taille du tableau.

La valeur rendue

La quantité de paramètres d'entrée de l'indicateur avec le handle indiqué dans le cas de l'erreur rend-1. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Exemple:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {

//---la quantité de fenêtres sur le graphique (il y a toujours quand même une fenêtre
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
//--- passons selon les fenêtres du graphique
   for(int w=0;w<windows;w++)
     {
      //---combien d'indicateurs sont dans cette fenêtre/sous- fenêtre,
      int total=ChartIndicatorsTotal(0,w);
      //--- trierons tous les indicateurs dans la fenêtre
      for(int i=0;i<total;i++)
        {
         //--- recevrons le nom court de l'indicateur
         string name=ChartIndicatorName(0,w,i);
         //--- recevrons le handle de l'indicateur
         int handle=ChartIndicatorGet(0,w,name);
```

© 2000-2015, MetaQuotes Software Corp.

```
         //--- déduisons dans un journal
         PrintFormat("Window=%d,  indicator #%d,  handle=%d",w,i,handle);
         //---
         MqlParam parameters[];
         ENUM_INDICATOR indicator_type;
         int params=IndicatorParameters(handle,indicator_type,parameters);
         //--- le titre du message
         string par_info="Short name "+name+", type "
                         +EnumToString(ENUM_INDICATOR(indicator_type))+"\r\n";
         //---
         for(int p=0;p<params;p++)
           {
            par_info+=StringFormat("parameter %d: type=%s, long_value=%d, double_value
                                   p,
                                   EnumToString((ENUM_DATATYPE)parameters[p].type),
                                   parameters[p].integer_value,
                                   parameters[p].double_value,
                                   parameters[p].string_value
                                   );
           }
         Print(par_info);
        }
      //--- ont passé selon tous les indicateurs dans la fenêtre
     }
//---
  }
```

Voir aussi

ChartIndicatorGet()

© 2000-2015, MetaQuotes Software Corp.

IndicatorRelease

Supprime le handle de l'indicateur et libère la partie de calcul de l'indicateur, si personne ne se sert plus d'elle.

```
bool  IndicatorRelease(
   int       indicator_handle,     // handle de l'indicateur
   );
```

La valeur rendue

Rend true en cas du succès, autrement false.

Note

La fonction permet de supprimer le handle de l'indicateur, si on n'a pas plus besoin de lui, et ainsi permet d'économiser la mémoire. On supprime le handle tout de suite, on supprime la partie de calcul dans quelque temps (s'il n'y a pas plus d'appel à elle).

Exemple:

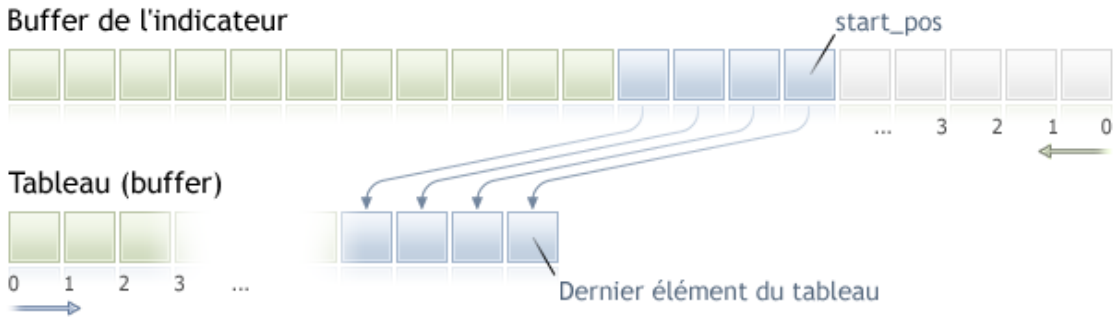
© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                        Test_IndicatorRelease.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- input parameters
input int                MA_Period=15;
input int                MA_shift=0;
input ENUM_MA_METHOD     MA_smooth=MODE_SMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
//--- garderons le handle de l'indicateur
int MA_handle;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---  créons le handle de l'indicateur
   MA_handle=iMA(Symbol(),0,MA_Period,MA_shift,MA_smooth,price);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- si dans la variable globale la valeur n'est pas encore donnée
   if(GlobalVariableCheck("MA_value")==0)
     {
      //--- tableau dynamique pour la réception des valeurs de l'indicateur
      double v[];
      //--- recevrons les valeurs de l'indicateur sur deux dernières barres
      if(CopyBuffer(MA_handle,0,0,2,v)==2 && v[1]!=EMPTY_VALUE)
        {
         //--- retiendrons la valeur sur la barre avant-dernière dans la variable glob
         if(GlobalVariableSet("MA_value",v[1]))
           {
            //--- libérons le handle de l'indicateur
            if(!IndicatorRelease(MA_handle))
               Print("IndicatorRelease() failed. Error ",GetLastError());
           }
        }
     }
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

CopyBuffer

Reçoit des données d'un tampon spécifiée de l'indicateur sur la quantité nécessaire au tableau "buffer".



Le compte des éléments des données copiées (le tampon d'indicateur avec l'index buffer_num) de la position de départ est compté du présent au passé, c'est-à-dire la position de départ est égale à 0, signifie la barre courante (la valeur de l'indicateur pour la barre courante).

Au copiage de nombre inconnu à l'avance de données à titre du tableau-récepteur buffer[] il est désirable d'utiliser le tableau dynamique, puisque la fonction CopyBuffer() tâche de distribuer la grandeur du tableau acceptant sous la grandeur des données copiées. Si à titre du tableau-récepteur buffer[] est le tampon d'indicateur (le tableau, préalablement fixé pour le stockage des valeurs de l'indicateur par la fonction SetIndexBufer()), on admet le copiage partiel. On peut regarder l'exemple dans l'indicateur d'utilisateur Awesome_Oscillator.mq5 de la livraison standard du terminal.

S'il est nécessaire de produire le copiage partiel des valeurs de l'indicateur à un autre tableau (le tampon pas indicateur), pour ces buts il est nécessaire d'utiliser le tableau intermédiaire, auquel on copie le nombre demandé. Et déjà de ce tableau-intermédiaire produire le copiage par élément du nombre nécessaire de valeurs aux espaces nécessaires du tableau acceptant.

S'il est nécessaire de copier le nombre connu d'avance de données, il vaut mieux faire cela au tampon extrait statiquement, pour éviter d'allocation de mémoire excessive.

N'importe quelle propriété a le tableau de réception - as_series=true ou as_series=false, les données seront copiées pour que l'élément le plus vieux par le temps sera au début de la mémoire physique, allouée pour le tableau. Il y a 3 variantes de la fonction. Il y a 3 variantes de la fonction.

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyBuffer(
   int       indicator_handle,     // handle de l'indicateur
   int       buffer_num,           // numéro du tampon de l'indicateur
   int       start_pos,            // d'où commencerons
   int       count,                // combien nous copions
   double    buffer[]              // tableau, où les données serront copier
   );
```

L'appel selon la date initiale et le nombre d'éléments demandés

```
int  CopyBuffer(
   int       indicator_handle,     // handle de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
   int       buffer_num,          // numéro du tampon de l'indicateur
   datetime  start_time,          // de quelle date
   int       count,               // combien nous copions
   double    buffer[]             // tableau, où les données serront copier
   );
```

L'appel selon les dates initiales et finales de l'intervalle demandé du temps

```
int  CopyBuffer(
   int       indicator_handle,    // handle de l'indicateur
   int       buffer_num,          // numéro du tampon de l'indicateur
   datetime  start_time,          // de quelle date
   datetime  stop_time,           // combien nous copions
   double    buffer[]             // tableau, où les données serront copier
   );
```

Paramètres

indicator_handle

[in] Le handle de l'indicateur, reçu par la fonction correspondante d'indicateur.

buffer_num

[in] Le numéro du tampon d'indicateur.

start_pos

[in] Le numéro du premier élément copié.

count

[in] Le nombre d'éléments copiés.

start_time

[in] Le temps de la barre correspondante au premier élément.

stop_time

[in] Le temps de la barre correspondante au dernier élément.

buffer[]

[out] Le tableau du type double.

La valeur rendue

Le nombre d'éléments de tableau copiés ou-1 en cas de l'erreur.

Note

A la demande des données de l'indicateur, si les séries temporelles demandées ne sont pas encore construites ou il faut les télécharger du serveur, la fonction rendra tout de suite-1, mais de plus le procès du chargement/de la construction sera initié.

A la demande des données de l'expert ou le scénario, on initie le chargement du serveur, si le terminal n'a pas ces données localement, ou commencera la construction nécessaire de la série temporelle si on peut construire les données de l'histoire locale, mais ils ne sont pas encore prêts. La fonction rendra ce nombre de données, qui seront prêts au moment de l'expiration de timeout.

© 2000-2015, MetaQuotes Software Corp.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                TestCopyBuffer3.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input bool                AsSeries=true;
input int                 period=15;
input ENUM_MA_METHOD      smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE  price=PRICE_CLOSE;
input int                 shift=0;
//--- indicator buffers
double                    MABuffer[];
int                       ma_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Paramètre AsSeries = ",AsSeries);
   Print("Le tampon d'indicateur après SetIndexBuffer() est la série temporelle = ",
         ArrayGetAsSeries(MABuffer));
//--- set short indicator name
   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//--- set AsSeries(dependes from input parameter)
   ArraySetAsSeries(MABuffer,AsSeries);
   Print("Le tampon d'indicateur après ArraySetAsSeries(MABuffer,true); est la série t
         ArrayGetAsSeries(MABuffer));
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- check if all data calculated
   if(BarsCalculated(ma_handle)<rates_total) return(0);
//--- we can copy not all data
   int to_copy;
   if(prev_calculated>rates_total || prev_calculated<=0) to_copy=rates_total;
   else
     {
      to_copy=rates_total-prev_calculated;
      //--- last value is always copied
      to_copy++;
     }
//--- try to copy
   if(CopyBuffer(ma_handle,0,0,to_copy,MABuffer)<=0) return(0);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Dans l'exemple cité ci-dessus on illustre le remplissage du tampon d'indicateur par les valeurs de l'autre tampon d'indicateur de l'indicateur sur le même symbole/période.

L'exemple le plus complet de la demande des données historiques regardez dans le paragraphe Moyens du rattachement des objets. Dans le script, on montre comment recevoir la valeur de l'indicateur iFractals sur les dernières 1000 barres et comment ensuite afficher sur le graphique les dix derniers fractales en haut et en bas. On peut utiliser une telle méthode pour tous les indicateurs, qui ont les espaces des valeurs et se réalisent d'habitude à l'aide des styles suivants de la construction:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2015, MetaQuotes Software Corp.

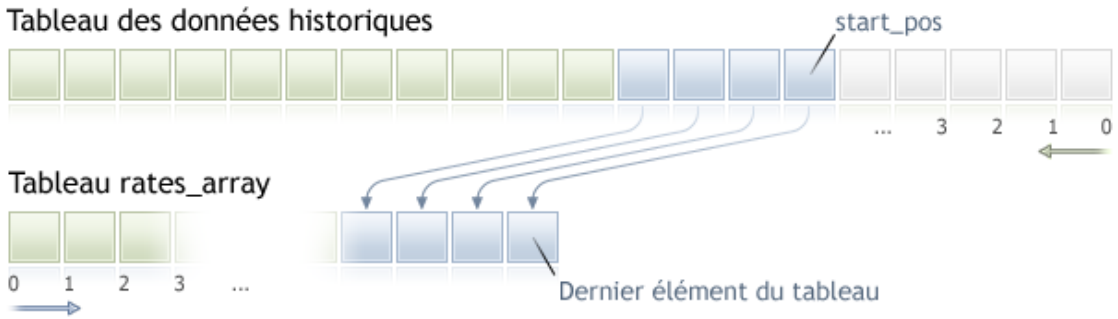
Voir aussi

Les propriétés des indicateurs d'utilisateur, SetIndexBuffer

© 2000-2015, MetaQuotes Software Corp.

CopyRates

Reçoit au tableau rates_array les données historiques de la structure MqlRates du symbole-la période indiqué en quantité indiquée. Le compte des éléments de la position de départ est conduit du présent au passé, c'est-à-dire position de départ est égale à 0, signifie la barre courante.



Au copiage de nombre inconnu à l'avance de données il est désirable d'utiliser le tableau dynamique comme le tableau) récepteur, puisque si des données sont moins (ou plus), que contient le tableau, on produit la tentative de la redistribution du tableau pour que les données demandées se placent entièrement.

S'il est nécessaire de copier le nombre connu d'avance de données, il vaut mieux faire cela au tampon extrait statiquement, pour éviter d'allocation de mémoire excessive.

Ce n'est pas grave, quelle propriété a le tableau de réception - as_series=true ou as_series=false, les données seront copiées de telle sorte que l'élément le plus vieux par le temps soit au début de mémoire physique allouée pour le tableau. Il y a 3 variantes de la fonction.

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyRates(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  temps trame,         // période
   int              start_pos,         // d'où commencerons
   int              count,             // combien nous copions
   MqlRates         rates_array[]      // tableau, où les données serront copier
   );
```

L'appel selon la date initiale et le nombre d'éléments demandés

```
int  CopyRates(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  temps trame,         // période
   datetime         start_time,        // de quelle date
   int              count,             // combien nous copions
   MqlRates         rates_array[]      // tableau, où les données serront copier
   );
```

L'appel selon les dates initiales et finales de l'intervalle demandé du temps

```
int  CopyRates(
```

© 2000-2015, MetaQuotes Software Corp.

```
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,         // période
   datetime         start_time,        // de quelle date
   datetime         stop_time,         // jusqu'à quelle date
   MqlRates         rates_array[]      // tableau, où les données serront copier
   );
```

Paramètres

symbol_name

[in] Le symbole.

temps_trame

[in] La période.

start_time

[in] Le temps de la barre correspondante au premier élément.

start_pos

[in] Le numéro du premier élément copié.

count

[in] Le nombre d'éléments copiés.

stop_time

[in] Le temps de la barre correspondante au dernier élément.

rates_array[]

[out] Le tableau du type MqlRates.

La valeur rendue

Le nombre d'éléments de tableau copiés ou-1 en cas de l'erreur.

Note

Si l'interligne des données demandées se trouve entièrement en dehors des données accessibles sur le serveur, la fonction rend -1. Dans le cas où on demande les données en dehors de TERMINAL_MAXBARS (le nombre maximal des barres sur le graphique), la fonction rendra aussi-1.

A la demande des données de l'indicateur, si les séries temporelles demandées ne sont pas encore construites ou il faut les télécharger du serveur, la fonction rendra tout de suite-1, mais de plus le procès du chargement/de la construction sera initié.

A la demande des données de l'expert ou le scénario, on initie le chargement du serveur, si le terminal n'a pas ces données localement, ou commencera la construction nécessaire de la série temporelle si on peut construire les données de l'histoire locale, mais ils ne sont pas encore prêts. La fonction rendra ce nombre de données, qui seront prêts au moment de l'expiration de timeout, mais le chargement de l'histoire se prolongera et à la demande suivante analogique la fonction rendra déjà plus de données.

A la demande des données selon la date initiale et la quantité d'éléments demandés reviennent seulement les données, dont la date est moins (avant) ou est égale à l'indiquée. En cela l'intervalle est spécifié et il est pris en considération à près la seconde. C'est-à-dire la date de l'ouverture de

© 2000-2015, MetaQuotes Software Corp.

n'importe quelle barre, pour laquelle revient la valeur (le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time), est toujours égale ou moins de la date indiquée.

A la demande des données dans la gamme donnée des dates reviennent seulement les données qui se trouvent dans l'interligne demandé, de plus l'interligne est spécifiée et est prise en considération à près la seconde. C'est-à-dire, le temps de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur, (Le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time) se trouve toujours dans l'interligne demandé.

Ainsi, si le jour courant de la semaine est le Samedi, à la tentative de copier les données sur le temps trame d'une semaine avec l'indication *start_time= Le dernier Mardi* et *stop_time= Le dernier Vendredi* la fonction rendra 0, puisque le temps de l'ouverture sur le temps trame d'une semaine incombe toujours pour le dimanche, mais aucune barre d'une semaine ne se trouve pas dans la gamme indiquée.

S'il est nécessaire de recevoir la valeur correspondante à la barre courante inachevée, on peut utiliser la première forme de l'appel avec l'indication *start_pos*=0 et *count*=1.

Exemple:

```
void OnStart()
  {
//---
   MqlRates rates[];
   ArraySetAsSeries(rates,true);
   int copied=CopyRates(Symbol(),0,0,100,rates);
   if(copied>0)
     {
      Print("Est copié des barres: "+copied);
      string format="open = %G, high = %G, low = %G, close = %G, volume = %d";
      string out;
      int size=fmin(copied,10);
      for(int i=0;i<size;i++)
        {
         out=i+":"+TimeToString(rates[i].time);
         out=out+" "+StringFormat(format,
                                  rates[i].open,
                                  rates[i].high,
                                  rates[i].low,
                                  rates[i].close,
                                  rates[i].tick_volume);
         Print(out);
        }
     }
   else Print("On n'a pas réussi à recevoir les données historiques selon le symbole '
  }
```

L'exemple le plus complet de la demande des données historiques regardez dans le paragraphe Moyens du rattachement des objets. Dans le script, on montre comment recevoir la valeur de l'indicateur iFractals sur les dernières 1000 barres et comment ensuite afficher sur le graphique les dix derniers

© 2000-2015, MetaQuotes Software Corp.

fractales en haut et en bas. On peut utiliser une telle méthode pour tous les indicateurs, qui ont les espaces des valeurs et se réalisent d'habitude à l'aide des styles suivants de la construction:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

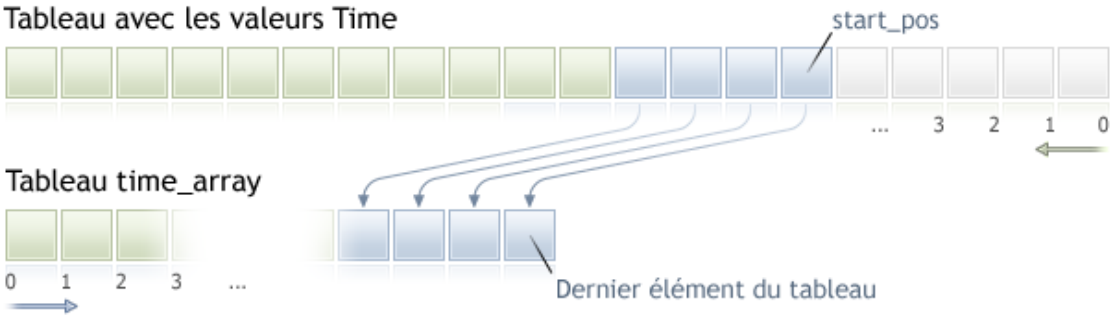
Voir aussi

Les structures et les classes, TimeToString, StringFormat

© 2000-2015, MetaQuotes Software Corp.

CopyTime

La fonction reçoit au tableau time_array les données historiques du temps de l'ouverture des barres pour la paire indiquée le caractère-la période en quantité indiquée. Il est nécessaire de noter que le compte des éléments de la position de départ est conduit du présent au passé, c'est-à-dire position de départ est égale à 0, signifie la barre courante.



Au copiage de nombre inconnu à l'avance de données il est désirable d'utiliser le tableau dynamique comme le tableau) récepteur, puisque si des données sont moins (ou plus), que contient le tableau, on produit la tentative de la redistribution du tableau pour que les données demandées se placent entièrement.

S'il est nécessaire de copier le nombre connu d'avance de données, il vaut mieux faire cela au tampon extrait statiquement, pour éviter d'allocation de mémoire excessive.

Ce n'est pas grave, quelle propriété a le tableau de réception - as_series=true ou as_series=false, les données seront copiées de telle sorte que l'élément le plus vieux par le temps soit au début de mémoire physique allouée pour le tableau. Il y a 3 variantes de la fonction.

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyTime(
   string           symbol_name,     // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,       // période
   int              start_pos,       // d'où commencerons
   int              count,           // combien nous copions
   datetime         time_array[]     // tableau pour le copiage du temps de l'ouvertu
   );
```

L'appel selon la date initiale et le nombre d'éléments demandés

```
int  CopyTime(
   string           symbol_name,     // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,       // période
   datetime         start_time,      // de quelle date
   int              count,           // combien nous copions
   datetime         time_array[]     // tableau pour le copiage du temps de l'ouvertu
   );
```

L'appel selon les dates initiales et finales de l'intervalle demandé du temps

© 2000-2015, MetaQuotes Software Corp.

```
int  CopyTime(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   datetime         start_time,       // de quelle date
   datetime         stop_time,        // jusqu'à quelle date
   datetime         time_array[]      // tableau pour le copiage du temps de l'ouvertu
   );
```

Paramètres

symbol_name

[in] Le symbole.

temps_trame

[in] La période.

start_pos

[in] Le numéro du premier élément copié.

count

[in] Le nombre d'éléments copiés.

start_time

[in] Le temps de la barre correspondante au premier élément.

stop_time

[in] Le temps de la barre correspondante au dernier élément.

time_array[]

[out] Le tableau du type datetime.

La valeur rendue

Le nombre d'éléments de tableau copiés ou-1 en cas de l'erreur.

Note

Si l'interligne des données demandées se trouve entièrement en dehors des données accessibles sur le serveur, la fonction rend -1. Dans le cas où on demande les données en dehors de TERMINAL_MAXBARS (le nombre maximal des barres sur le graphique), la fonction rendra aussi-1.

A la demande des données de l'indicateur, si les séries temporelles demandées ne sont pas encore construites ou il faut les télécharger du serveur, la fonction rendra tout de suite-1, mais de plus le procès du chargement/de la construction sera initié.

A la demande des données de l'expert ou le scénario, on initie le chargement du serveur, si le terminal n'a pas ces données localement, ou commencera la construction nécessaire de la série temporelle si on peut construire les données de l'histoire locale, mais ils ne sont pas encore prêts. La fonction rendra ce nombre de données, qui seront prêts au moment de l'expiration de timeout, mais le chargement de l'histoire se prolongera et à la demande suivante analogique la fonction rendra déjà plus de données.

© 2000-2015, MetaQuotes Software Corp.

A la demande des données selon la date initiale et la quantité d'éléments demandés reviennent seulement les données, dont la date est moins (avant) ou est égale à l'indiquée. En cela l'intervalle est spécifié et il est pris en considération à près la seconde. C'est-à-dire la date de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur (le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time), est toujours égale ou moins de la date indiquée.

A la demande des données dans la gamme donnée des dates reviennent seulement les données qui se trouvent dans l'interligne demandé, de plus l'interligne est spécifiée et est prise en considération à près la seconde. C'est-à-dire, le temps de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur, (Le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time) se trouve toujours dans l'interligne demandé.

Ainsi, si le jour courant de la semaine est le Samedi, à la tentative de copier les données sur le temps trame d'une semaine avec l'indication *start_time= Le dernier Mardi* et *stop_time= Le dernier Vendredi* la fonction rendra 0, puisque le temps de l'ouverture sur le temps trame d'une semaine incombe toujours pour le dimanche, mais aucune barre d'une semaine ne se trouve pas dans la gamme indiquée.

S'il est nécessaire de recevoir la valeur correspondante à la barre courante inachevée, on peut utiliser la première forme de l'appel avec l'indication *start_pos*=0 et *count*=1.

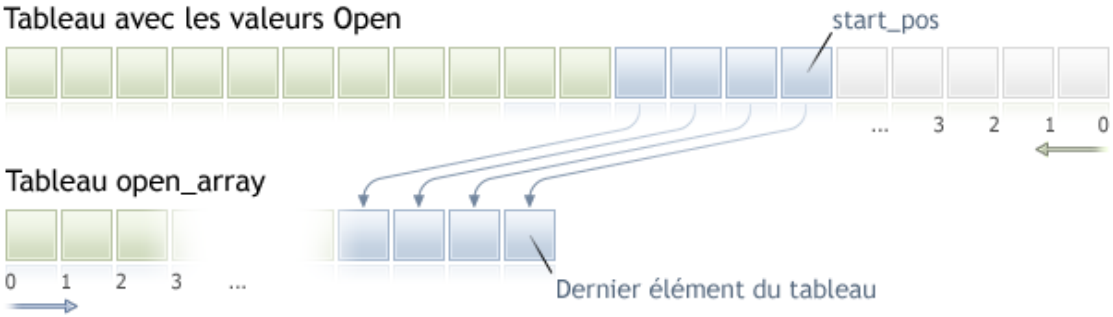
L'exemple le plus complet de la demande des données historiques regardez dans le paragraphe Moyens du rattachement des objets. Dans le script, on montre comment recevoir la valeur de l'indicateur iFractals sur les dernières 1000 barres et comment ensuite afficher sur le graphique les dix derniers fractales en haut et en bas. On peut utiliser une telle méthode pour tous les indicateurs, qui ont les espaces des valeurs et se réalisent d'habitude à l'aide des styles suivants de la construction:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2015, MetaQuotes Software Corp.

CopyOpen

La fonction reçoit au tableau open_array les données historiques du temps de l'ouverture des barres pour la paire indiquée le caractère-la période en quantité indiquée. Il est nécessaire de noter que le compte des éléments de la position de départ est conduit du présent au passé, c'est-à-dire position de départ est égale à 0, signifie la barre courante.



Au copiage de nombre inconnu à l'avance de données il est désirable d'utiliser le tableau dynamique comme le tableau) récepteur, puisque si des données sont moins (ou plus), que contient le tableau, on produit la tentative de la redistribution du tableau pour que les données demandées se placent entièrement.

S'il est nécessaire de copier le nombre connu d'avance de données, il vaut mieux faire cela au tampon extrait statiquement, pour éviter d'allocation de mémoire excessive.

Ce n'est pas grave, quelle propriété a le tableau de réception - as_series=true ou as_series=false, les données seront copiées de telle sorte que l'élément le plus vieux par le temps soit au début de mémoire physique allouée pour le tableau. Il y a 3 variantes de la fonction..

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyOpen(
   string            symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES   temps_trame,        // période
   int               start_pos,        // d'où commencerons
   int               count,            // combien nous copions
   double            open_array[]      // tableau pour le copiage des prix de l'ouvertu
   );
```

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyOpen(
   string            symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES   temps_trame,        // période
   datetime          start_time,       // de quelle date
   int               count,            // combien nous copions
   double            open_array[]      // tableau pour le copiage des prix de l'ouvertu
   );
```

L'appel selon les dates initiales et finales de l'intervalle demandé du temps

© 2000-2015, MetaQuotes Software Corp.

```
int  CopyOpen(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   datetime         start_time,       // de quelle date
   datetime         stop_time,        // jusqu'à quelle date
   double           open_array[]      // tableau pour le copiage des prix de l'ouvertu
   );
```

Paramètres

symbol_name

[in] Le symbole.

temps_trame

[in] La période.

start_pos

[in] Le numéro du premier élément copié.

count

[in] Le nombre d'éléments copiés.

start_time

[in] Le temps de la barre correspondante au premier élément.

stop_time

[in] Le temps de la barre correspondante au dernier élément.

open_array[]

[out] Le tableau du type double.

La valeur rendue

Le nombre d'éléments de tableau copiés ou-1 en cas de l'erreur.

Note

Si l'interligne des données demandées se trouve entièrement en dehors des données accessibles sur le serveur, la fonction rend -1. Dans le cas où on demande les données en dehors de TERMINAL_MAXBARS (le nombre maximal des barres sur le graphique), la fonction rendra aussi-1.

A la demande des données de l'indicateur, si les séries temporelles demandées ne sont pas encore construites ou il faut les télécharger du serveur, la fonction rendra tout de suite-1, mais de plus le procès du chargement/de la construction sera initié.

A la demande des données de l'expert ou le scénario, on initie le chargement du serveur, si le terminal n'a pas ces données localement, ou commencera la construction nécessaire de la série temporelle si on peut construire les données de l'histoire locale, mais ils ne sont pas encore prêts. La fonction rendra ce nombre de données, qui seront prêts au moment de l'expiration de timeout, mais le chargement de l'histoire se prolongera et à la demande suivante analogique la fonction rendra déjà plus de données.

A la demande des données selon la date initiale et la quantité d'éléments demandés reviennent seulement les données, dont la date est moins (avant) ou est égale à l'indiquée. En cela l'intervalle

© 2000-2015, MetaQuotes Software Corp.

est spécifié et il est pris en considération à près la seconde. C'est-à-dire la date de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur (le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time), est toujours égale ou moins de la date indiquée.

A la demande des données dans la gamme donnée des dates reviennent seulement les données qui se trouvent dans l'interligne demandé, de plus l'interligne est spécifiée et est prise en considération à près la seconde. C'est-à-dire, le temps de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur, (Le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time) se trouve toujours dans l'interligne demandé.

Ainsi, si le jour courant de la semaine est le Samedi, à la tentative de copier les données sur le temps trame d'une semaine avec l'indication *start_time= Le dernier Mardi* et *stop_time= Le dernier Vendredi* la fonction rendra 0, puisque le temps de l'ouverture sur le temps trame d'une semaine incombe toujours pour le dimanche, mais aucune barre d'une semaine ne se trouve pas dans la gamme indiquée.

S'il est nécessaire de recevoir la valeur correspondante à la barre courante inachevée, on peut utiliser la première forme de l'appel avec l'indication *start_pos*=0 et *count*=1.

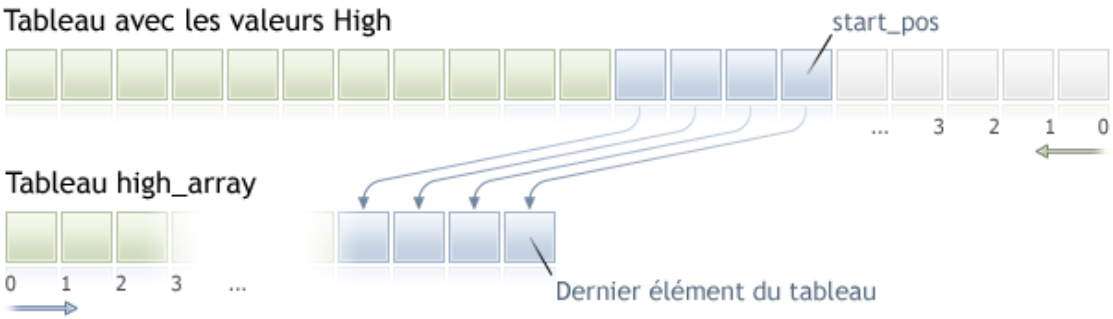
L'exemple de la demande des données historiques regardez dans le paragraphe Moyens du rattachement des objets. Dans le script, on montre comment recevoir la valeur de l'indicateur iFractals sur les dernières 1000 barres et comment ensuite afficher sur le graphique les dix derniers fractales en haut et en bas. On peut utiliser une telle méthode pour tous les indicateurs, qui ont les espaces des valeurs et se réalisent d'habitude à l'aide des styles suivants de la construction:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2015, MetaQuotes Software Corp.

CopyHigh

La fonction reçoit au tableau high_array les données historiques des prix maximum des barres pour la paire indiquée le caractère-la période en quantité indiquée. Il est nécessaire de noter que le compte des éléments de la position de départ est conduit du présent au passé, c'est-à-dire position de départ est égale à 0, signifie la barre courante.



Au copiage de nombre inconnu à l'avance de données il est désirable d'utiliser le tableau dynamique comme le tableau) récepteur, puisque si des données sont moins (ou plus), que contient le tableau, on produit la tentative de la redistribution du tableau pour que les données demandées se placent entièrement.

S'il est nécessaire de copier le nombre connu d'avance de données, il vaut mieux faire cela au tampon extrait statiquement, pour éviter d'allocation de mémoire excessive.

Ce n'est pas grave, quelle propriété a le tableau de réception - as_series=true ou as_series=false, les données seront copiées de telle sorte que l'élément le plus vieux par le temps soit au début de mémoire physique allouée pour le tableau. Il y a 3 variantes de la fonction

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyHigh(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   int              start_pos,        // d'où commencerons
   int              count,            // combien nous copions
   double           high_array[]      // tableau pour le copiage des prix maximum
   );
```

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyHigh(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   datetime         start_time,       // de quelle date
   int              count,            // combien nous copions
   double           high_array[]      // tableau pour le copiage des prix maximum
   );
```

L'appel selon les dates initiales et finales de l'intervalle demandé du temps

© 2000-2015, MetaQuotes Software Corp.

```
int  CopyHigh(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps trame,        // période
   datetime         start_time,       // de quelle date
   datetime         stop_time,        // jusqu'à quelle date
   double           high_array[]      // tableau pour le copiage des prix maximum
   );
```

Paramètres

symbol_name

[in] Le symbole.

temps trame

[in] La période.

start_pos

[in] Le numéro du premier élément copié.

count

[in] Le nombre d'éléments copiés.

start_time

[in] Le temps de la barre correspondante au premier élément.

stop_time

[in] Le temps de la barre correspondante au dernier élément.

high_array[]

[out] Le tableau du type double.

La valeur rendue

Le nombre d'éléments de tableau copiés ou-1 en cas de l'erreur.

Note

Si l'interligne des données demandées se trouve entièrement en dehors des données accessibles sur le serveur, la fonction rend -1. Dans le cas où on demande les données en dehors de TERMINAL_MAXBARS (le nombre maximal des barres sur le graphique), la fonction rendra aussi-1.

A la demande des données de l'indicateur, si les séries temporelles demandées ne sont pas encore construites ou il faut les télécharger du serveur, la fonction rendra tout de suite-1, mais de plus le procès du chargement/de la construction sera initié.

A la demande des données de l'expert ou le scénario, on initie le chargement du serveur, si le terminal n'a pas ces données localement, ou commencera la construction nécessaire de la série temporelle si on peut construire les données de l'histoire locale, mais ils ne sont pas encore prêts. La fonction rendra ce nombre de données, qui seront prêts au moment de l'expiration de timeout, mais le chargement de l'histoire se prolongera et à la demande suivante analogique la fonction rendra déjà plus de données.

A la demande des données selon la date initiale et la quantité d'éléments demandés reviennent seulement les données, dont la date est moins (avant) ou est égale à l'indiquée. En cela l'intervalle

© 2000-2015, MetaQuotes Software Corp.

est spécifié et il est pris en considération à près la seconde. C'est-à-dire la date de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur (le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time), est toujours égale ou moins de la date indiquée.

A la demande des données dans la gamme donnée des dates reviennent seulement les données qui se trouvent dans l'interligne demandé, de plus l'interligne est spécifiée et est prise en considération à près la seconde. C'est-à-dire, le temps de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur, (Le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time) se trouve toujours dans l'interligne demandé.

Ainsi, si le jour courant de la semaine est le Samedi, à la tentative de copier les données sur le temps trame d'une semaine avec l'indication *start_time= Le dernier Mardi* et *stop_time= Le dernier Vendredi* la fonction rendra 0, puisque le temps de l'ouverture sur le temps trame d'une semaine incombe toujours pour le dimanche, mais aucune barre d'une semaine ne se trouve pas dans la gamme indiquée.

S'il est nécessaire de recevoir la valeur correspondante à la barre courante inachevée, on peut utiliser la première forme de l'appel avec l'indication *start_pos*=0 et *count*=1.

Exemple:

```
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Exemple de la sortie des valeurs High[i] et Low[i]"
#property description "pour les barres choisies éventuellement"

double High[],Low[];
//+------------------------------------------------------------------+
//| Recevrons Low pour le numéro spécifié de la barre                |
//+------------------------------------------------------------------+
double iLow(string symbol,ENUM_TIMEFRAMES temps trame,int index)
  {
   double low=0;
   ArraySetAsSeries(Low,true);
   int copied=CopyLow(symbol, temps trame, 0,Bars(symbol,timeframe),Low);
   if(copied>0 && index<copied) low=Low[index];
   return(low);
  }
//+------------------------------------------------------------------+
//| Recevrons High pour le numéro spécifié de la barre               |
//+------------------------------------------------------------------+
double iHigh(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double high=0;
   ArraySetAsSeries(High,true);
   int copied=CopyHigh(symbol,timeframe,0,Bars(symbol,timeframe),High);
   if(copied>0 && index<copied) high=High[index];
   return(high);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- affichons sur chaque tick de la valeur High et Low pour la barre avec l'index,
//--- égal à la seconde de l'entrée du tick
   datetime t=TimeCurrent();
   int sec=t%60;
   printf("High[%d] = %G  Low[%d] = %G",
          sec,iHigh(Symbol(),0,sec),
          sec,iLow(Symbol(),0,sec));
  }
```

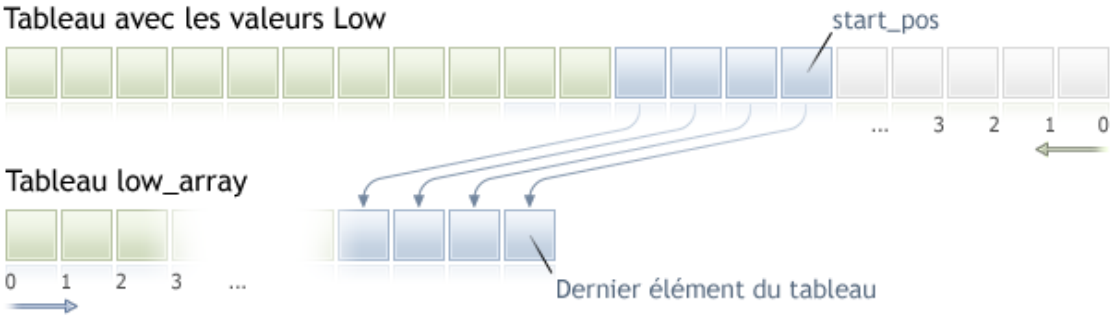
L'exemple de la demande des données historiques regardez dans le paragraphe Moyens du rattachement des objets. Dans le script, on montre comment recevoir la valeur de l'indicateur iFractals sur les dernières 1000 barres et comment ensuite afficher sur le graphique les dix derniers fractales en haut et en bas. On peut utiliser une telle méthode pour tous les indicateurs, qui ont les espaces des valeurs et se réalisent d'habitude à l'aide des styles suivants de la construction:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2015, MetaQuotes Software Corp.

CopyLow

La fonction reçoit au tableau low_array les données historiques des prix minimum des barres pour la paire indiquée le caractère-la période en quantité indiquée. Il est nécessaire de noter que le compte des éléments de la position de départ est conduit du présent au passé, c'est-à-dire position de départ est égale à 0, signifie la barre courante.



Au copiage de nombre inconnu à l'avance de données il est désirable d'utiliser le tableau dynamique comme le tableau) récepteur, puisque si des données sont moins (ou plus), que contient le tableau, on produit la tentative de la redistribution du tableau pour que les données demandées se placent entièrement.

S'il est nécessaire de copier le nombre connu d'avance de données, il vaut mieux faire cela au tampon extrait statiquement, pour éviter d'allocation de mémoire excessive.

Ce n'est pas grave, quelle propriété a le tableau de réception - as_series=true ou as_series=false, les données seront copiées de telle sorte que l'élément le plus vieux par le temps soit au début de mémoire physique allouée pour le tableau. Il y a 3 variantes de la fonction

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyLow(
   string            symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES   temps_trame,        // période
   int               start_pos,        // d'où commencerons
   int               count,            // combien nous copions
   double            low_array[]       // tableau pour le copiage des prix minimum
   );
```

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyLow(
   string            symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES   temps_trame,        // période
   datetime          start_time,       // de quelle date
   int               count,            // combien nous copions
   double            low_array[]       // tableau pour le copiage des prix minimum
   );
```

L'appel selon les dates initiales et finales de l'intervalle demandé du temps

© 2000-2015, MetaQuotes Software Corp.

```
int  CopyLow(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   datetime         start_time,       // de quelle date
   datetime         stop_time,        // jusqu'à quelle date
   double           low_array[]       // tableau pour le copiage des prix minimum
   );
```

Paramètres

symbol_name

[in] Le symbole.

temps_trame

[in] La période.

start_pos

[in] Le numéro du premier élément copié.

count

[in] Le nombre d'éléments copiés.

start_time

[in] Le temps de la barre correspondante au premier élément.

stop_time

[in] Le temps de la barre correspondante au dernier élément.

low_array[]

[out] Le tableau du type double.

La valeur rendue

Le nombre d'éléments de tableau copiés ou-1 en cas de l'erreur.

Note

Si l'interligne des données demandées se trouve entièrement en dehors des données accessibles sur le serveur, la fonction rend -1. Dans le cas où on demande les données en dehors de TERMINAL_MAXBARS (le nombre maximal des barres sur le graphique), la fonction rendra aussi-1.

A la demande des données de l'indicateur, si les séries temporelles demandées ne sont pas encore construites ou il faut les télécharger du serveur, la fonction rendra tout de suite-1, mais de plus le procès du chargement/de la construction sera initié.

A la demande des données de l'expert ou le scénario, on initie le chargement du serveur, si le terminal n'a pas ces données localement, ou commencera la construction nécessaire de la série temporelle si on peut construire les données de l'histoire locale, mais ils ne sont pas encore prêts. La fonction rendra ce nombre de données, qui seront prêts au moment de l'expiration de timeout, mais le chargement de l'histoire se prolongera et à la demande suivante analogique la fonction rendra déjà plus de données.

A la demande des données selon la date initiale et la quantité d'éléments demandés reviennent seulement les données, dont la date est moins (avant) ou est égale à l'indiquée. En cela l'intervalle

© 2000-2015, MetaQuotes Software Corp.

est spécifié et il est pris en considération à près la seconde. C'est-à-dire la date de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur (le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time), est toujours égale ou moins de la date indiquée.

A la demande des données dans la gamme donnée des dates reviennent seulement les données qui se trouvent dans l'interligne demandé, de plus l'interligne est spécifiée et est prise en considération à près la seconde. C'est-à-dire, le temps de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur, (Le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time) se trouve toujours dans l'interligne demandé.

Ainsi, si le jour courant de la semaine est le Samedi, à la tentative de copier les données sur le temps trame d'une semaine avec l'indication *start_time= Le dernier Mardi* et *stop_time= Le dernier Vendredi* la fonction rendra 0, puisque le temps de l'ouverture sur le temps trame d'une semaine incombe toujours pour le dimanche, mais aucune barre d'une semaine ne se trouve pas dans la gamme indiquée.

S'il est nécessaire de recevoir la valeur correspondante à la barre courante inachevée, on peut utiliser la première forme de l'appel avec l'indication *start_pos*=0 et *count*=1.

L'exemple de la demande des données historiques regardez dans le paragraphe Moyens du rattachement des objets. Dans le script, on montre comment recevoir la valeur de l'indicateur iFractals sur les dernières 1000 barres et comment ensuite afficher sur le graphique les dix derniers fractales en haut et en bas. On peut utiliser une telle méthode pour tous les indicateurs, qui ont les espaces des valeurs et se réalisent d'habitude à l'aide des styles suivants de la construction:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

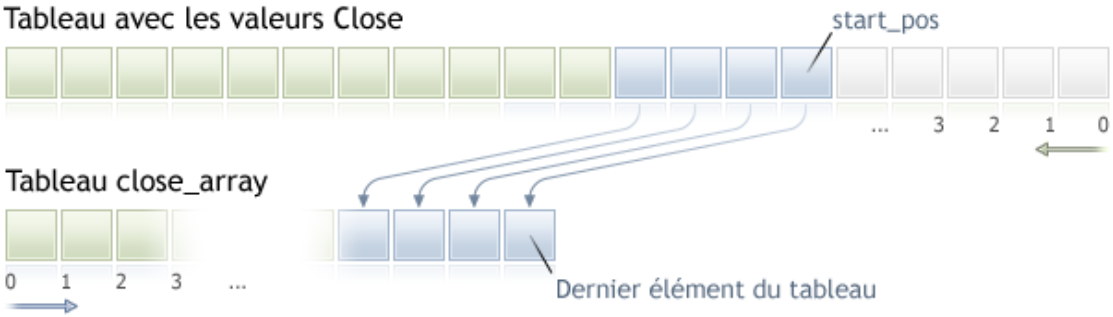
Voir aussi

CopyHigh

© 2000-2015, MetaQuotes Software Corp.

CopyClose

La fonction reçoit au tableau close_array les données historiques des prix de la clôture des barres pour la paire indiquée le caractère-la période en quantité indiquée. Il est nécessaire de noter que le compte des éléments de la position de départ est conduit du présent au passé, c'est-à-dire position de départ est égale à 0, signifie la barre courante.



Au copiage de nombre inconnu à l'avance de données il est désirable d'utiliser le tableau dynamique comme le tableau) récepteur, puisque si des données sont moins (ou plus), que contient le tableau, on produit la tentative de la redistribution du tableau pour que les données demandées se placent entièrement.

S'il est nécessaire de copier le nombre connu d'avance de données, il vaut mieux faire cela au tampon extrait statiquement, pour éviter d'allocation de mémoire excessive.

Ce n'est pas grave, quelle propriété a le tableau de réception - as_series=true ou as_series=false, les données seront copiées de telle sorte que l'élément le plus vieux par le temps soit au début de mémoire physique allouée pour le tableau. Il y a 3 variantes de la fonction

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyClose(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,         // période
   int              start_pos,         // d'où commencerons
   int              count,             // combien nous copions
   double           close_array[]      // tableau pour le copiage des prix de la clôtu
   );
```

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyClose(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,         // période
   datetime         start_time,        // de quelle date
   int              count,             // combien nous copions
   double           close_array[]      //tableau pour le copiage des prix de la clôtur
   );
```

L'appel selon les dates initiales et finales de l'intervalle demandé du temps

© 2000-2015, MetaQuotes Software Corp.

```
int  CopyClose(
   string           symbol_name,       // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,         // période
   datetime         start_time,        // de quelle date
   datetime         stop_time,         // jusqu'à quelle date
   double           close_array[]      // tableau pour le copiage des prix de la clôtu
   );
```

Paramètres

symbol_name

[in] Le symbole.

temps_trame

[in] La période.

start_pos

[in] Le numéro du premier élément copié.

count

[in] Le nombre d'éléments copiés.

start_time

[in] Le temps de la barre correspondante au premier élément.

stop_time

[in] Le temps de la barre correspondante au dernier élément.

close_array[]

[out] Le tableau du type double.

La valeur rendue

Le nombre d'éléments de tableau copiés ou-1 en cas de l'erreur.

Note

Si l'interligne des données demandées se trouve entièrement en dehors des données accessibles sur le serveur, la fonction rend -1. Dans le cas où on demande les données en dehors de TERMINAL_MAXBARS (le nombre maximal des barres sur le graphique), la fonction rendra aussi-1.

A la demande des données de l'indicateur, si les séries temporelles demandées ne sont pas encore construites ou il faut les télécharger du serveur, la fonction rendra tout de suite-1, mais de plus le procès du chargement/de la construction sera initié.

A la demande des données de l'expert ou le scénario, on initie le chargement du serveur, si le terminal n'a pas ces données localement, ou commencera la construction nécessaire de la série temporelle si on peut construire les données de l'histoire locale, mais ils ne sont pas encore prêts. La fonction rendra ce nombre de données, qui seront prêts au moment de l'expiration de timeout, mais le chargement de l'histoire se prolongera et à la demande suivante analogique la fonction rendra déjà plus de données.

A la demande des données selon la date initiale et la quantité d'éléments demandés reviennent seulement les données, dont la date est moins (avant) ou est égale à l'indiquée. En cela l'intervalle

© 2000-2015, MetaQuotes Software Corp.

est spécifié et il est pris en considération à près la seconde. C'est-à-dire la date de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur (le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time), est toujours égale ou moins de la date indiquée.

A la demande des données dans la gamme donnée des dates reviennent seulement les données qui se trouvent dans l'interligne demandé, de plus l'interligne est spécifiée et est prise en considération à près la seconde. C'est-à-dire, le temps de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur, (Le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time) se trouve toujours dans l'interligne demandé.

Ainsi, si le jour courant de la semaine est le Samedi, à la tentative de copier les données sur le temps trame d'une semaine avec l'indication *start_time= Le dernier Mardi* et *stop_time= Le dernier Vendredi* la fonction rendra 0, puisque le temps de l'ouverture sur le temps trame d'une semaine incombe toujours pour le dimanche, mais aucune barre d'une semaine ne se trouve pas dans la gamme indiquée.

S'il est nécessaire de recevoir la valeur correspondante à la barre courante inachevée, on peut utiliser la première forme de l'appel avec l'indication *start_pos*=0 et *count*=1.

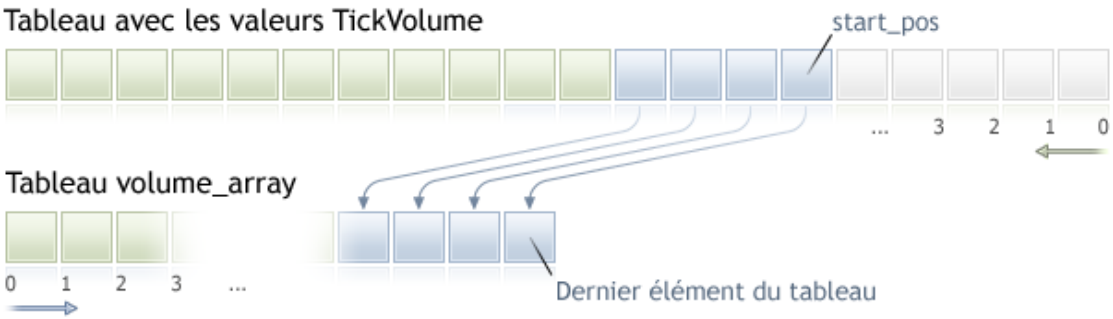
L'exemple de la demande des données historiques regardez dans le paragraphe Moyens du rattachement des objets. Dans le script, on montre comment recevoir la valeur de l'indicateur iFractals sur les dernières 1000 barres et comment ensuite afficher sur le graphique les dix derniers fractales en haut et en bas. On peut utiliser une telle méthode pour tous les indicateurs, qui ont les espaces des valeurs et se réalisent d'habitude à l'aide des styles suivants de la construction:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2015, MetaQuotes Software Corp.

CopyTickVolume

La fonction reçoit au tableau volume_array les données historiques des volumes de tick pour la paire indiquée le caractère-la période en quantité indiquée. Il est nécessaire de noter que le compte des éléments de la position de départ est conduit du présent au passé, c'est-à-dire position de départ est égale à 0, signifie la barre courante.



Au copiage de nombre inconnu à l'avance de données il est désirable d'utiliser le tableau dynamique comme le tableau) récepteur, puisque si des données sont moins (ou plus), que contient le tableau, on produit la tentative de la redistribution du tableau pour que les données demandées se placent entièrement.

S'il est nécessaire de copier le nombre connu d'avance de données, il vaut mieux faire cela au tampon extrait statiquement, pour éviter d'allocation de mémoire excessive.

Ce n'est pas grave, quelle propriété a le tableau de réception - as_series=true ou as_series=false, les données seront copiées de telle sorte que l'élément le plus vieux par le temps soit au début de mémoire physique allouée pour le tableau. Il y a 3 variantes de la fonction

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyTickVolume(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   int              start_pos,        // d'où commencerons
   int              count,            // combien nous copions
   long             volume_array[]    // tableau pour le copiage des volumes de tick
   );
```

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyTickVolume(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   datetime         start_time,       // de quelle date
   int              count,            // combien nous copions
   long             volume_array[]    // tableau pour le copiage des volumes de tick
   );
```

L'appel selon les dates initiales et finales de l'intervalle demandé du temps

© 2000-2015, MetaQuotes Software Corp.

```
int  CopyTickVolume(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   datetime         start_time,       // de quelle date
   datetime         stop_time,        // jusqu'à quelle date
   long             volume_array[]    // tableau pour le copiage des volumes de tick
   );
```

Paramètres

symbol_name

[in] Le symbole.

temps_trame

[in] La période.

start_pos

[in] Le numéro du premier élément copié.

count

[in] Le nombre d'éléments copiés.

start_time

[in] Le temps de la barre correspondante au premier élément.

stop_time

[in] Le temps de la barre correspondante au dernier élément.

volume_array[]

[out] Le tableau du type long.

La valeur rendue

Le nombre d'éléments de tableau copiés ou-1 en cas de l'erreur.

Note

Si l'interligne des données demandées se trouve entièrement en dehors des données accessibles sur le serveur, la fonction rend -1. Dans le cas où on demande les données en dehors de TERMINAL_MAXBARS (le nombre maximal des barres sur le graphique), la fonction rendra aussi-1.

A la demande des données de l'indicateur, si les séries temporelles demandées ne sont pas encore construites ou il faut les télécharger du serveur, la fonction rendra tout de suite-1, mais de plus le procès du chargement/de la construction sera initié.

A la demande des données de l'expert ou le scénario, on initie le chargement du serveur, si le terminal n'a pas ces données localement, ou commencera la construction nécessaire de la série temporelle si on peut construire les données de l'histoire locale, mais ils ne sont pas encore prêts. La fonction rendra ce nombre de données, qui seront prêts au moment de l'expiration de timeout, mais le chargement de l'histoire se prolongera et à la demande suivante analogique la fonction rendra déjà plus de données.

A la demande des données selon la date initiale et la quantité d'éléments demandés reviennent seulement les données, dont la date est moins (avant) ou est égale à l'indiquée. En cela l'intervalle

© 2000-2015, MetaQuotes Software Corp.

est spécifié et il est pris en considération à près la seconde. C'est-à-dire la date de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur (le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time), est toujours égale ou moins de la date indiquée.

A la demande des données dans la gamme donnée des dates reviennent seulement les données qui se trouvent dans l'interligne demandé, de plus l'interligne est spécifiée et est prise en considération à près la seconde. C'est-à-dire, le temps de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur, (Le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time) se trouve toujours dans l'interligne demandé.

Ainsi, si le jour courant de la semaine est le Samedi, à la tentative de copier les données sur le temps trame d'une semaine avec l'indication *start_time= Le dernier Mardi* et *stop_time= Le dernier Vendredi* la fonction rendra 0, puisque le temps de l'ouverture sur le temps trame d'une semaine incombe toujours pour le dimanche, mais aucune barre d'une semaine ne se trouve pas dans la gamme indiquée.

S'il est nécessaire de recevoir la valeur correspondante à la barre courante inachevée, on peut utiliser la première forme de l'appel avec l'indication *start_pos*=0 et *count*=1.

Exemple:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot TickVolume
#property indicator_label1  "TickVolume"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  C'143,188,139'
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      bars=3000;
//--- indicator buffers
double         TickVolumeBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,TickVolumeBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                  const double &open[],
                  const double &high[],
                  const double &low[],
                  const double &close[],
                  const long &tick_volume[],
                  const long &volume[],
                  const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      long timeseries[];
      ArraySetAsSeries(timeseries,true);
      int prices=CopyTickVolume(Symbol(),0,0,bars,timeseries);
      for(int i=0;i<rates_total-prices;i++) TickVolumeBuffer[i]=0.0;
      for(int i=0;i<prices;i++) TickVolumeBuffer[rates_total-1-i]=timeseries[prices-1-
      Print("On a reçu les valeurs historiques TickVolume: "+prices);
     }
   else
     {
      long timeseries[];
      int prices=CopyTickVolume(Symbol(),0,0,1,timeseries);
      TickVolumeBuffer[rates_total-1]=timeseries[0];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

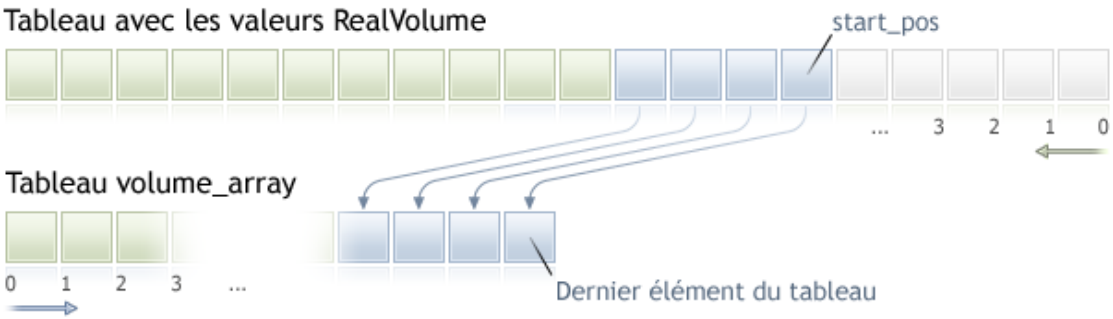
L'exemple le plus complet de la demande des données historiques regardez dans le paragraphe Moyens du rattachement des objets. Dans le script, on montre comment recevoir la valeur de l'indicateur iFractals sur les dernières 1000 barres et comment ensuite afficher sur le graphique les dix derniers fractales en haut et en bas. On peut utiliser une telle méthode pour tous les indicateurs, qui ont les espaces des valeurs et se réalisent d'habitude à l'aide des styles suivants de la construction:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2015, MetaQuotes Software Corp.

CopyRealVolume

La fonction reçoit au tableau volume_array les données historiques des volumes de commerce pour la paire indiquée le caractère-la période en quantité indiquée. Il est nécessaire de noter que le compte des éléments de la position de départ est conduit du présent au passé, c'est-à-dire position de départ est égale à 0, signifie la barre courante.



Au copiage de nombre inconnu à l'avance de données il est désirable d'utiliser le tableau dynamique comme le tableau) récepteur, puisque si des données sont moins (ou plus), que contient le tableau, on produit la tentative de la redistribution du tableau pour que les données demandées se placent entièrement.

S'il est nécessaire de copier le nombre connu d'avance de données, il vaut mieux faire cela au tampon extrait statiquement, pour éviter d'allocation de mémoire excessive.

Ce n'est pas grave, quelle propriété a le tableau de réception - as_series=true ou as_series=false, les données seront copiées de telle sorte que l'élément le plus vieux par le temps soit au début de mémoire physique allouée pour le tableau. Il y a 3 variantes de la fonction.

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyRealVolume(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   int              start_pos,        // d'où commencerons
   int              count,            // combien nous copions
   long             volume_array[]    // tableau pour le copiage des volumes
   );
```

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopyRealVolume(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   datetime         start_time,       // de quelle date
   int              count,            // combien nous copions
   long             volume_array[]    // tableau pour le copiage des volumes
   );
```

L'appel selon les dates initiales et finales de l'intervalle demandé du temps

© 2000-2015, MetaQuotes Software Corp.

```
int  CopyRealVolume(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps trame,        // période
   datetime         start_time,       // de quelle date
   datetime         stop_time,        // jusqu'à quelle date
   long             volume_array[]    // tableau pour le copiage des volumes
   );
```

Paramètres

symbol_name

[in] Le symbole.

temps trame

[in] La période.

start_pos

[in] Le numéro du premier élément copié.

count

[in] Le nombre d'éléments copiés.

start_time

[in] Le temps de la barre correspondante au premier élément.

stop_time

[in] Le temps de la barre correspondante au dernier élément.

volume_array[]

[out] Le tableau du type long.

La valeur rendue

Le nombre d'éléments de tableau copiés ou-1 en cas de l'erreur.

Note

Si l'interligne des données demandées se trouve entièrement en dehors des données accessibles sur le serveur, la fonction rend -1. Dans le cas où on demande les données en dehors de TERMINAL_MAXBARS (le nombre maximal des barres sur le graphique), la fonction rendra aussi-1.

A la demande des données de l'indicateur, si les séries temporelles demandées ne sont pas encore construites ou il faut les télécharger du serveur, la fonction rendra tout de suite-1, mais de plus le procès du chargement/de la construction sera initié.

A la demande des données de l'expert ou le scénario, on initie le chargement du serveur, si le terminal n'a pas ces données localement, ou commencera la construction nécessaire de la série temporelle si on peut construire les données de l'histoire locale, mais ils ne sont pas encore prêts. La fonction rendra ce nombre de données, qui seront prêts au moment de l'expiration de timeout, mais le chargement de l'histoire se prolongera et à la demande suivante analogique la fonction rendra déjà plus de données.

A la demande des données selon la date initiale et la quantité d'éléments demandés reviennent seulement les données, dont la date est moins (avant) ou est égale à l'indiquée. En cela l'intervalle

© 2000-2015, MetaQuotes Software Corp.

est spécifié et il est pris en considération à près la seconde. C'est-à-dire la date de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur (le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time), est toujours égale ou moins de la date indiquée.

A la demande des données dans la gamme donnée des dates reviennent seulement les données qui se trouvent dans l'interligne demandé, de plus l'interligne est spécifiée et est prise en considération à près la seconde. C'est-à-dire, le temps de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur, (Le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time) se trouve toujours dans l'interligne demandé.

Ainsi, si le jour courant de la semaine est le Samedi, à la tentative de copier les données sur le temps trame d'une semaine avec l'indication *start_time= Le dernier Mardi* et *stop_time= Le dernier Vendredi* la fonction rendra 0, puisque le temps de l'ouverture sur le temps trame d'une semaine incombe toujours pour le dimanche, mais aucune barre d'une semaine ne se trouve pas dans la gamme indiquée.

S'il est nécessaire de recevoir la valeur correspondante à la barre courante inachevée, on peut utiliser la première forme de l'appel avec l'indication *start_pos*=0 et *count*=1.

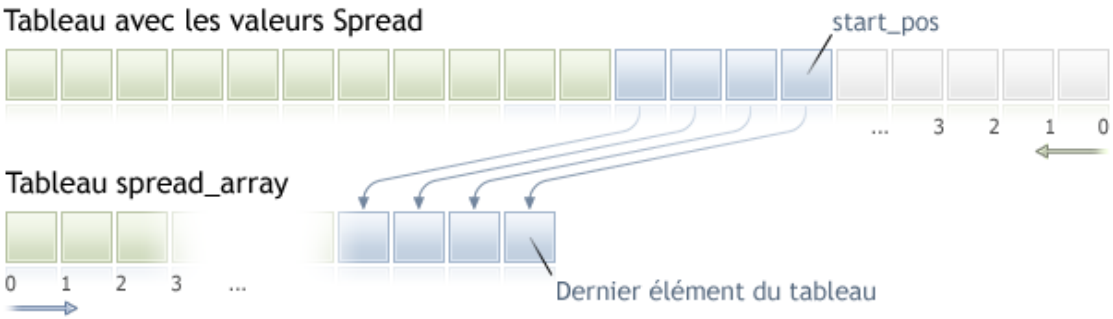
L'exemple de la demande des données historiques regardez dans le paragraphe Moyens du rattachement des objets. Dans le script, on montre comment recevoir la valeur de l'indicateur iFractals sur les dernières 1000 barres et comment ensuite afficher sur le graphique les dix derniers fractales en haut et en bas. On peut utiliser une telle méthode pour tous les indicateurs, qui ont les espaces des valeurs et se réalisent d'habitude à l'aide des styles suivants de la construction:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2015, MetaQuotes Software Corp.

CopySpread

La fonction reçoit au tableau spread_array les données historiques des spread pour la paire indiquée le caractère-la période en quantité indiquée. Il est nécessaire de noter que le compte des éléments de la position de départ est conduit du présent au passé, c'est-à-dire position de départ est égale à 0, signifie la barre courante.



Au copiage de nombre inconnu à l'avance de données il est désirable d'utiliser le tableau dynamique comme le tableau-récepteur, puisque si des données sont moins (ou plus), que contient le tableau, on produit la tentative de la redistribution du tableau pour que les données demandées se placent entièrement.

S'il est nécessaire de copier le nombre connu d'avance de données, il vaut mieux faire cela au tampon extrait statiquement, pour éviter d'allocation de mémoire excessive.

Ce n'est pas grave, quelle propriété a le tableau de réception - as_series=true ou as_series=false, les données seront copiées de telle sorte que l'élément le plus vieux par le temps soit au début de mémoire physique allouée pour le tableau. Il y a 3 variantes de la fonction

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopySpread(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   int              start_pos,        // d'où commencerons
   int              count,            // combien nous copions
   int              spread_array[]    // tableau pour le copiage des spread
   );
```

L'appel à la position initiale et du nombre d'éléments demandés

```
int  CopySpread(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   datetime         start_time,       // de quelle date
   int              count,            // combien nous copions
   int              spread_array[]    // tableau pour le copiage des spread
   );
```

L'appel selon les dates initiales et finales de l'intervalle demandé du temps

© 2000-2015, MetaQuotes Software Corp.

```
int  CopySpread(
   string           symbol_name,      // nom du symbole
   ENUM_TIMEFRAMES  temps_trame,        // période
   datetime         start_time,       // de quelle date
   datetime         stop_time,        // jusqu'à quelle date
   int              spread_array[]    // tableau pour le copiage des spread
   );
```

Paramètres

symbol_name

[in] Le symbole.

temps_trame

[in] La période

start_pos

[in] Le numéro du premier élément copié.

count

[in] Le nombre d'éléments copiés.

start_time

[in] Le temps de la barre correspondante au premier élément.

stop_time

[in] Le temps de la barre correspondante au dernier élément.

spread_array[]

[out] Le tableau du type int.

La valeur rendue

Le nombre d'éléments de tableau copiés ou-1 en cas de l'erreur.

Note

Si l'interligne des données demandées se trouve entièrement en dehors des données accessibles sur le serveur, la fonction rend -1. Dans le cas où on demande les données en dehors de TERMINAL_MAXBARS (le nombre maximal des barres sur le graphique), la fonction rendra aussi-1.

A la demande des données de l'indicateur, si les séries temporelles demandées ne sont pas encore construites ou il faut les télécharger du serveur, la fonction rendra tout de suite-1, mais de plus le procès du chargement/de la construction sera initié.

A la demande des données de l'expert ou le scénario, on initie le chargement du serveur, si le terminal n'a pas ces données localement, ou commencera la construction nécessaire de la série temporelle si on peut construire les données de l'histoire locale, mais ils ne sont pas encore prêts. La fonction rendra ce nombre de données, qui seront prêts au moment de l'expiration de timeout, mais le chargement de l'histoire se prolongera et à la demande suivante analogique la fonction rendra déjà plus de données.

A la demande des données selon la date initiale et la quantité d'éléments demandés reviennent seulement les données, dont la date est moins (avant) ou est égale à l'indiquée. En cela l'intervalle

© 2000-2015, MetaQuotes Software Corp.

est spécifié et il est pris en considération à près la seconde. C'est-à-dire la date de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur (le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time), est toujours égale ou moins de la date indiquée.

A la demande des données dans la gamme donnée des dates reviennent seulement les données qui se trouvent dans l'interligne demandé, de plus l'interligne est spécifiée et est prise en considération à près la seconde. C'est-à-dire, le temps de l'ouverture de n'importe quelle barre, pour laquelle revient la valeur, (Le volume, le spread, la valeur dans le tampon d'indicateur, le prix Open, High, Low, Close ou le temps de l'ouverture Time) se trouve toujours dans l'interligne demandé.

Ainsi, si le jour courant de la semaine est le Samedi, à la tentative de copier les données sur le temps trame d'une semaine avec l'indication *start_time= Le dernier Mardi* et *stop_time= Le dernier Vendredi* la fonction rendra 0, puisque le temps de l'ouverture sur le temps trame d'une semaine incombe toujours pour le dimanche, mais aucune barre d'une semaine ne se trouve pas dans la gamme indiquée.

S'il est nécessaire de recevoir la valeur correspondante à la barre courante inachevée, on peut utiliser la première forme de l'appel avec l'indication *start_pos*=0 et *count*=1.

Exemple:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Spread
#property indicator_label1  "Spread"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      bars=3000;
//--- indicator buffers
double         SpreadBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,SpreadBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                  const double &open[],
                  const double &high[],
                  const double &low[],
                  const double &close[],
                  const long &tick_volume[],
                  const long &volume[],
                  const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      int spread_int[];
      ArraySetAsSeries(spread_int,true);
      int spreads=CopySpread(Symbol(),0,0,bars,spread_int);
      Print("On a reçu les valeurs historiques du spread: ",spreads);
      for (int i=0;i<spreads;i++)
      {
      SpreadBuffer[rates_total-1-i]=spread_int[i];
      if(i<=30) Print("spread["+i+"] = ",spread_int[i]);
      }
     }
   else
     {
      double Ask,Bid;
      Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
      Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
      Comment("Ask = ",Ask,"  Bid = ",Bid);
      SpreadBuffer[rates_total-1]=(Ask-Bid)/Point();
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

L'exemple le plus complet de la demande des données historiques regardez dans le paragraphe Moyens du rattachement des objets. Dans le script, on montre comment recevoir la valeur de l'indicateur iFractals sur les dernières 1000 barres et comment ensuite afficher sur le graphique les dix derniers fractales en haut et en bas. On peut utiliser une telle méthode pour tous les indicateurs, qui ont les espaces des valeurs et se réalisent d'habitude à l'aide des styles suivants de la construction:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2015, MetaQuotes Software Corp.

CopyTicks

Функция получает в массив ticks_array тики, накопленные терминалом за текущую рабочую сессию. Необходимо отметить, что порядок индексации ведется от прошлого к настоящему, то есть тик с индексом 0 является самым старым в массиве.

```
int  CopyTicks(
   string           symbol_name,           // имя символа
   MqlTick&         ticks_array[],         // массив для приема тиков
   uint             flags=COPY_TICKS_ALL,  // флаг, определяющий тип получаемых тиков
   ulong            from=0,                // дата, начиная с которой запрашиваются ти
   uint             count=0                // количество последних тиков, которые необ
   );
```

Параметры

symbol_name

[in] Символ.

ticks_array

[out] Массив типа MqlTick для приема тиков.

flags

[in] Флаг, определяющий тип получаемых тиков. COPY_TICKS_INFO - только Bid и Ask, COPY_TICKS_TRADE - только Last и Volume, COPY_TICKS_ALL - все тики.

from

[in] Дата, начиная с которой запрашиваются тики. Указывается в миллисекундах с 01.01.1970.

count

[in] Количество запрашиваемых тиков. Если дата и количество не указаны, передаются все доступные тики, но не более 2000 последних тиков.

Возвращаемое значение

Количество скопированных тиков либо -1 в случае ошибки.

Примечание

Функция OnTick() не является обработчиком каждого тика, она уведомляет эксперта об изменениях на рынке. Изменения могут быть пакетными: в терминал может одновременно прийти несколько тиков, но функция OnTick() будет вызвана лишь один раз для уведомления эксперта о последнем состоянии рынка. Функция CopyTicks() позволяет запрашивать и анализировать все пришедшие тики.

Если параметры *from* и *count* не указаны, то в массив *ticks_array[]* будут записаны все доступные тики, но не более 2000. Параметр flags позволяет задать тип требуемых тиков.

Пример:

```
//--- input parameters
input int      ticks=10;  // количество запрашиваемых тиков
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- массив для приема тиков
   MqlTick tick_array[];
//--- запросим тики
   int copied=CopyTicks(_Symbol,tick_array,COPY_TICKS_ALL,0,ticks);
//--- если тики получены, то выведем на график значения Bid и Ask
   if(copied>0)
     {
      string comment="#  Time       Bid        Ask\r\n";
      //--- сформируем содержимое комментария
      for(int i=0;i<copied;i++)
        {
         MqlTick tick=tick_array[i];
         string tick_string=StringFormat("%d: %s  %G  %G",
                                         i,
                                         TimeToString(tick.time,TIME_MINUTES|TIME_SEC
                                         tick.bid,
                                         tick.ask);
         comment=comment+tick_string+"\r\n";
        }
        //--- выводим комментарий на график
        Comment(comment);
     }
   else // сообщим об ошибке при получении тиков
     {
      Comment("Не удалось загрузить тики. GetLastError()=",GetLastError());
     }
  }
```

Смотри также

SymbolInfoTick, Структура для получения текущих цен, OnTick()

© 2000-2015, MetaQuotes Software Corp.

Les opérations avec les graphiques

Les fonctions pour le travail avec les graphiques. Toutes les opérations avec les graphiques sont employées seulement dans les experts et les scripts.

Les fonctions établissant les propriétés du graphique, servent en fait pour envoyer à lui des commandes pour le changement. À l'exécution réussie de ces fonctions la commande se trouve dans une file d'attente des événements du graphique. Le changement du graphique est produit en train du traitement de la file d'attente des événements du graphique donné.

Pour cette raison il ne faut pas attendre la mise à jour visuelle immédiate du graphique après l'appel des fonctions données. En général, la mise à jour du graphique est produite par le terminal automatiquement selon les événements du changement - l'entrée de la nouvelle cotation, le changement de la taille de la fenêtre du graphique etc.

Pour la mise à jour forcée de l'apparence du graphique utilisez la commande pour redessiner le graphiqueChartRedraw().

Fonction	Action
ChartApplyTemplate	Applique au graphique indiqué le modèle du fichier indiqué
ChartSaveTemplate	Enregistre les paramètres actuels du graphique dans un modèle avec le nom spécifié
ChartWindowFind	Rend le numéro de sous- fenêtre, dans laquelle il y a un indicateur
ChartTimePriceToXY	Convertit les coordonnées du graphique de la représentation graphique du temps/prix pour les axes X et Y
ChartXYToTimePrice	Convertit les coordonnées X et Y du graphique en valeurs - le temps et le prix
ChartOpen	Ouvre un nouveau graphique avec le symbole indiqué et la période
ChartClose	Ferme le graphique indiqué
ChartFirst	Rend l'identificateur du premier graphique du terminal de client
ChartNext	Rend l'identificateur du graphique, suivant de l'indiqué
ChartSymbol	Rend le nom du symbole du graphique indiqué
ChartPeriod	Rend la valeur de la période du graphique indiqué
ChartRedraw	Appelle la copie forcée du graphique indiqué
ChartSetDouble	Spécifie la valeur du type double de la propriété correspondante du graphique indiqué

© 2000-2015, MetaQuotes Software Corp.

ChartSetInteger	Spécifie la valeur du type entier (datetime, int, color, bool ou char) de la propriété correspondante du graphique indiqué
ChartSetString	Spécifie la valeur du type string de la propriété correspondante du graphique indiqué
ChartGetDouble	Rend la valeur de la propriété correspondante du graphique indiqué
ChartGetInteger	Rend la valeur entière de la propriété correspondante du graphique indiqué
ChartGetString	Rend la valeur de chaîne de la propriété correspondante du graphique indiqué
ChartNavigate	Réalise le décalage du graphique indiqué au nombre indiqué de barres par rapport à la position indiquée du graphique
ChartID	Rend l'identificateur du graphique courant
ChartIndicatorAdd	Ajoute l'indicateur avec le handle indiqué sur la fenêtre indiquée du graphique
ChartIndicatorDelete	Supprime l'indicateur avec le nom spécifié de la fenêtre indiquée du graphique
ChartIndicatorGet	Rend le handle de l'indicateur avec le nom court indiqué sur la fenêtre indiquée du graphique
ChartIndicatorName	Rend le nom court de l'indicateur selon le numéro dans la liste des indicateurs sur la fenêtre indiquée du graphique
ChartIndicatorsTotal	Rend le nombre de tous indicateurs qui sont attachés à la fenêtre indiquée du graphique
ChartWindowOnDropped	Rend le numéro de sous-fenêtre du graphique, sur laquelle on jette l'expert donné, le script, l'objet ou l'indicateur par la souris
ChartPriceOnDropped	Rend la coordonnée de prix correspondant au point, dans laquelle on jette par la souris l'expert donné ou le script
ChartTimeOnDropped	Rend la coordonnée temporaire correspondant au point, dans laquelle on jette par la souris l'expert donné ou le script
ChartXOnDropped	Rend la coordonnée selon l'axe X, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris
ChartYOnDropped	Rend la coordonnée selon l'axe Y, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris

© 2000-2015, MetaQuotes Software Corp.

ChartSetSymbolPeriod	Change les valeurs du symbole et la période du graphique indiqué
ChartScreenShot	Fournit un screenshot du graphique de son état actuel dans un format de GIF, PNG ou BMP en fonction de l'extension spécifiée

© 2000-2015, MetaQuotes Software Corp.

ChartApplyTemplate

Applique au graphique le modèle indiqué. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartApplyTemplate(
   long          chart_id,     // identificateur du graphique
   const string  filename      // nom du fichier avec le modèle
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

filename

[in] Le nom du fichier qui contient le modèle.

La valeur rendue

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError().

Note

L'expert sera déchargé et ne pourra pas continuer le travail si un nouveau modèle sera chargé de l'expert sur le graphique à qui il est adjoint au moyen de cette fonction.

В целях безопасности право на торговлю при применении шаблона к графику может ограничиваться:

> **Права на торговлю не могут быть повышены при запуске советника путем применения шаблона с помощью функции ChartApplyTemplate().**

Если у mql5-программы, которая вызывает функцию ChartApplyTemplate(), отсутствуют права на торговлю, то эксперт, загруженный при помощи шаблона, также не будет иметь прав на торговлю вне зависимости от настроек шаблона.

Если у mql5-программы, которая вызывает функцию ChartApplyTemplate(), есть права на торговлю, а в настройках шаблона права отсутствуют, то советник, загруженный при помощи шаблона, не будет иметь прав на торговлю.

Utilisation des modèles

Par les moyens du langage MQL5 on peut spécifier la multitude de propriétés du graphique, ainsi qu'établir les couleurs à l'aide de la fonction ChartSetInteger() :

- La couleur du fond du graphique;
- La couleur des axes, de l'échelle et de la ligne OHLC;
- La couleur de la grille;
- La couleur des volumes et des niveaux de l'ouverture des positions;
- La couleur de la barre en haut, de l'ombre et de l'encadrement du corps de la bougie d'haussier;

© 2000-2015, MetaQuotes Software Corp.

- La couleur de la barre en bas, de l'ombre et de l'encadrement du corps de la bougie de baissier;
- La couleur de la ligne du graphique et des chandeliers japonais "Doji";
- La couleur du corps de la bougie d'haussier;
- La couleur du corps de la bougie de baissier;
- La couleur de la ligne du prix Bid;
- La couleur de la ligne du prix Ask;
- La couleur de la ligne du prix du dernier marché passé (Last);
- La couleur des niveaux des ordres Stop (Stop Loss et Take Profit).

En outre la multituded'objets graphiques et des indicateurspeut être sur le graphique. Il suffit de configurer une fois l'apparence du graphique avec tous les indicateurs nécessaires et le garder comme un modèle pour appliquer ce modèle à n'importe quel graphique plus tard.

La fonction ChartApplyTemplate () est destinée à être utilisé par le modèle précédemment gardé et peut être utilisée dans n'importe quel programme mql5. Le chemin vers le fichier qui contient le modèle est transmis comme le deuxième paramètre de la fonction ChartApplyTemplate (). est transmis le chemin vers le fichier qui contient le modèle. La recherche du fichier du modèle se réalise selon les règles suivantes:

- si au début du chemin il y a le séparateur - la barre oblique inversée "\" (est écrit comme "\\"), le modèle est recherché relativement le chemin - le répertoire_des données_du terminal\MQL5,
- s'il n'y a pas de la barre oblique inversée, alors le modèle est recherché par rapport le fichier EX5 exécutable où passe l'appel de la fonction (ChartApplyTemplate);
- si le modèle n'est pas trouvé dans les deux premiers variants, la recherche est effectuée dans un dossier le répertoire_du terminal\Profiles\Templates\.

Ici le répertoire_du terminal signifie le dossier à partir duquel a été lancé le terminal de client MetaTrader 5 et le répertoire_des données_du terminal signifie le dossier qui contient les fichiers modifiés et son emplacement peut dépendre du type du système d'exploitation et du nom d'utilisateur et des paramètres de sécurité de l'ordinateur. En général, ce sont les dossiers différents, bien que dans certains cas ils peuvent coïncider.

On peut savoir les emplacements des dossiers le répertoire_des données_du terminal et le répertoire_du terminal à l'aide de la fonction TerminalInfoString().

```
//--- le répertoire d'où est lancé le terminal
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
   Print("le répertoire du terminal:",terminal_path);
//--- le répertoire des données du terminal, dans lequel se trouve le dossier MQL5 ave
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   Print("Le répertoire des données du terminal:",terminal_data_path);
```

Les exemples de l'enregistrement:

```
//--- cherchons le modèle dans le dossier le répertoire_des données_du terminal\MQL5\
ChartApplyTemplate(0,"\\first_template.tpl"))
//--- cherchons le modèle dans le dossier le répertoire_du fichier_EX5_exécutable\, ap
ChartApplyTemplate(0,"second_template.tpl"))
//--- cherchons le modèle dans le dossier le répertoire_du fichier_EX5_exécutable\My_t
ChartApplyTemplate(0,"My_templates\\third_template.tpl"))
```

Les modèles ne se rapportent pas aux ressources, on ne peut pas les insérer dans un fichier EX5

© 2000-2015, MetaQuotes Software Corp.

exécutable.

Exemple:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- пример применения шаблона, расположенного в каталоге \MQL5\Files
   if(FileIsExist("my_template.tpl"))
     {
      Print("Шаблон my_template.tpl найден в каталоге \Files'");
      //--- применим найденный шаблон
      if(ChartApplyTemplate(0,"\\Files\\my_template.tpl"))
        {
         Print("Применили успешно шаблон 'my_template.tpl'");
         //--- принудительно перерисуем график для быстрого показа изменений
         ChartRedraw();
        }
      else
         Print("Не удалось применить шаблон 'my_template.tpl', ошибка ",GetLastError()
     }
   else
     {
      Print("Файл 'my_template.tpl' не найден в папке "
            +TerminalInfoString(TERMINAL_PATH)+"\\MQL5\\Files");
     }
  }
```

Voir aussi

Ressources

© 2000-2015, MetaQuotes Software Corp.

ChartSaveTemplate

Garde les paramètres actuels de graphique dans le modèle avec le nom spécifié.

```
bool  ChartSaveTemplate(
   long          chart_id,     // l'identificateur du graphique
   const string  filename      // le nom du fichier pour la sauvegarde du modèle
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

filename

[in] Le nom du fichier pour la sauvegarde du modèle. L'extension ". tpl" sera ajouté au nom du fichier automatiquement, il ne faut pas le spécifier. Le modèle s'est gardé dans le dossier **le répertoire_du terminal**\Profiles\Templates\ et peut être utilisé pour l'application manuelle dans le terminal. Si le modèle avec ce nom existe déjà, son contenu sera enregistré de nouveau.

La valeur rendue

En cas de l'exécution successive la fonction rend true, autrement rend false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError () .

Note

Le modèle vous permet de garder les paramètres du graphique avec tous ses indicateurs et les objets graphiques, pour l'appliquer sur un autre graphique.

Exemple:

```
//+------------------------------------------------------------------+
//|                                      Test_ChartSaveTemplate.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- input parameters
input string               symbol="GBPUSD";  // le symbole d'un nouveau graphique
input ENUM_TIMEFRAMES      period=PERIOD_H3; // le timeframe d'un nouveau graphique
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tout d'abord attachons les indicateurs au graphique
   int handle;
//---préparons l'indicateur à l'utilisation
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!PrepareZigzag(NULL,0,handle)) return; // la tentative a échoué, sortons
//--- attachons l'indicateur sur le graphique actuel, mais dans une fenêtre séparée
   if(!ChartIndicatorAdd(0,1,handle))
     {
      PrintFormat("On n'a pas réussi à ajouter %s/%s l'indicateur avec le handle =%d s
                  _Symbol,
                  EnumToString(_Period),
                  handle,
                  GetLastError());
      //--- cessons avant terme le travail du programme
      return;
     }
//--- ordonnons au graphique de mettre à jour pour voir tout de suite l'indicateur att
   ChartRedraw();
//--- trouvons deux dernières fractures du zigzag
   double two_values[];
   datetime two_times[];
   if(!GetLastTwoFractures(two_values,two_times,handle))
     {
      PrintFormat("On n'a pas réussi à trouver deux dernières fractures dans l'indicat
      //--- cessons avant terme le travail du programme
      return;
     }
//--- maintenant attachons le canal de la divergence standard
   string channel="StdDeviation Channel";
   if(!ObjectCreate(0,channel,OBJ_STDDEVCHANNEL,0,two_times[1],0))
     {
      PrintFormat("On n'a pas réussi à créer l'objet %s. Code de l'erreur %d",
                  EnumToString(OBJ_STDDEVCHANNEL),GetLastError());
      return;
     }
   else
     {
      //--- le canal est créé, définissons le deuxième point d'appui
      ObjectSetInteger(0,channel,OBJPROP_TIME,1,two_times[0]);
      //--- spécifions le texte de l'infobulle au canal
      ObjectSetString(0,channel,OBJPROP_TOOLTIP,"Demo from MQL5 Help");
      //--- mettons à jour le graphique
      ChartRedraw();
     }
//--- sauvegardons le résultat dans un modèle
   ChartSaveTemplate(0,"StdDevChannelOnZigzag");
//--- ouvrons un nouveau graphique et y attachons un modèle sauvegardé
   long new_chart=ChartOpen(symbol,period);
   //--- activons la démonstration des infobulles émergentes pour les objets graphique
   ChartSetInteger(new_chart,CHART_SHOW_OBJECT_DESCR,true);
   if(new_chart!=0)
     {
      //--- attachons un modèle sauvegardé sur un nouveau graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
      ChartApplyTemplate(new_chart,"StdDevChannelOnZigzag");
     }
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| Créé le handle du zigzag et assure la disponibilité de ses données |
//+------------------------------------------------------------------+
bool PrepareZigzag(string sym,ENUM_TIMEFRAMES tf,int &h)
  {
   ResetLastError();
//--- l'indicateur Zigzag doit être dans le dossier le répertoire_des données_du termi
   h=iCustom(sym,tf,"Examples\\Zigzag");
   if(h==INVALID_HANDLE)
     {
      PrintFormat("%s: On n'a pas réussi à créer le handle de l'indicateur Zigzag. Coc
                  __FUNCTION__,GetLastError());
      return false;
     }
//--- pendant la création du handle de l'indicateur dont il aura besoin de temps pour
   int k=0; // la quantité de tentatives pour attendre le calcul de l'indicateur
//--- attendons le calcul dans la boucle, faisons la pause de 50 millisecondes, si le
   while(BarsCalculated(h)<=0)
     {
      k++;
      //--- déduisons le nombre de tentatives
      PrintFormat("%s: k=%d",__FUNCTION__,k);
 //--- attendons 50 millisecondes, alors que l'indicateur sera calculé
      Sleep(50);
 //--- Si est fait plus de 100 tentatives, alors quelque chose ne va pas
      if(k>100)
        {
         //--- informons sur le problème
         PrintFormat("On n'a pas réussi à calculer l'indicateur pour%d de tentatives!"
         //--- cessons avant terme le travail du programme
         return false;
        }
     }
//--- tout est prêt, l'indicateur est créé et les valeurs sont calculées
   return true;
  }
//+------------------------------------------------------------------+
//| Cherche deux dernières fractures du zigzag et place aux tableaux |
//+------------------------------------------------------------------+
bool GetLastTwoFractures(double &get_values[],datetime &get_times[],int handle)
  {
   double values[];          // le tableau pour la réception des valeurs du zigzag
   datetime times[];         // le tableau pour recevoir le temps
   int size=100;             // la taille des tableaux
   ResetLastError();
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- copions les dernières 100 valeurs de l'indicateur
   int copied=CopyBuffer(handle,0,0,size,values);
//--- vérifions combien on a été copié
   if(copied<100)
     {
      PrintFormat("%s: On n'a pas réussi à copier %d valeurs de l'indicateur avec le h
                  __FUNCTION__,size,handle,GetLastError());
      return false;
     }
//--- définissons l'ordre de l'accès au tableau comme dans une série temporelle
   ArraySetAsSeries(values,true);
//--- inscrivons ici les numéros des barres, sur lesquelles sont trouvés les fractures
   int positions[];
//--- spécifions les tailles des tableaux
   ArrayResize(get_values,3); ArrayResize(get_times,3); ArrayResize(positions,3);
//--- compteurs
   int i=0,k=0;
//--- commençons la recherche des fractures
   while(i<100)
     {
      double v=values[i];
      //--- les valeurs vides ne nous intéressent pas
      if(v!=0.0)
        {
         //--- rappelons le numéro de la barre
         positions[k]=i;
//--- rappelez la valeur du zigzag dans la fracture
         get_values[k]=values[i];
         PrintFormat("%s: Zigzag[%d]=%G",__FUNCTION__,i,values[i]);
         //--- augmentons le compteur
         k++;
         //--- si on a trouvé deux fractures, nous cassons la boucle
         if(k>2) break;
        }
      i++;
     }
//--- définissons l'ordre de l'accès aux tableaux comme dans une série temporelle
   ArraySetAsSeries(times,true);   ArraySetAsSeries(get_times,true);
   if(CopyTime(_Symbol,_Period,0,size,times)<=0)
     {
      PrintFormat("%s: On n'a pas réussi à copier %d valeurs de CopyTime(). Code de l'
                  __FUNCTION__,size,GetLastError());
      return false;
     }
//--- trouvons le temps de l'ouverture des barres, sur lesquelles il y avait 2 dernièr
   get_times[0]=times[positions[1]];// la valeur avant-dernière sera inscrite par la p
   get_times[1]=times[positions[2]];// la troisième valeur de la fin sera la deuxième
   PrintFormat("%s: le premier=%s, le deuxième=%s",__FUNCTION__,TimeToString(get_times
//--- tout a réussi
```

© 2000-2015, MetaQuotes Software Corp.

```
   return true;
  }
```

Voir aussi

ChartApplyTemplate(), Les ressources

© 2000-2015, MetaQuotes Software Corp.

ChartWindowFind

Rend le numéro de sous- fenêtre, dans laquelle il y a un indicateur. Il y a 2 variantes de la fonction.

1. La fonction cherche sur le graphique indiqué la sous- fenêtre avec l'indicateur indiqué par "le nom court" (le nom court est affiché à gauche au dessus de la sous- fenêtre) et cas du succès rend le numéro de la sous- fenêtre.

```
int  ChartWindowFind(
   long     chart_id,                  // identificateur du graphique
   string   indicator_shortname        // nom court de l'indicateur, regarde INDICATOR
```

2. La fonction doit être appélée de l'indicateur d'utilisateur et récupère le numéro de la sous- fenêtre, où cet indicateur travaille.

```
int  ChartWindowFind();
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

indicator_shortname

[in] Le nom court de l'indicateur.

La valeur rendue

Le numéro du sous- fenêtre en cas du succès. Le zéro signifie une principale fenêtre du graphique. En cas de l'échec rend -1.

Note

Si la deuxième variante de la fonction est appelée (sans paramètres) du scipt ou l'expert, elle rend-1.

Il ne faut pas confondre un nom court et de l'indicateur et le nom du fichier qui est spécifié pendant la création de l'indicateur par les fonctionsiCustom() et IndicatorCreate(). Si le nom court de l'indicateur n'est pas spécifié explicitement, alors pendant la compilation le nom du fichier avec le code initial de l'indicateur y est spécifié.

Il est nécessaire de former correctement un nom court de l'indicateur, qui s'écrit dans la propriété INDICATOR_SHORTNAME à l'aide de la fonctionIndicatorSetString () . Nous recommandons que le nom court contienne les valeurs des paramètres d'entrée de l'indicateur, puisque l'identification de l'indicateur supprimé du graphique dans la fonctionChartIndicatorDelete()est faite selon le nom court.

Exemple:

```
#property script_show_inputs
//--- input parameters
input string   shortName="MACD(12,26,9)";
//+------------------------------------------------------------------+
//| Rend le numéro de la fenêtre du graphique avec l'indicateur indiqué |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
int GetIndicatorSubWindowNumber(long chartID=0,string short_name="")
  {
   int window=-1;
//---
   if((ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR)
     {
      //--- fonction est appelée de l'indicateur, le nom n'est pas demandé
      window=ChartWindowFind();
     }
   else
     {
      //--- fonction est appelée de l'expert ou du script
      window=ChartWindowFind(0,short_name);
      if(window==-1) Print(__FUNCTION__+"(): Error = ",GetLastError());
     }
//---
   return(window);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int window=GetIndicatorSubWindowNumber(0,shortName);
   if(window!=-1)
      Print("Indicateur "+shortName+" se trouve dans la fenêtre #"+(string)window);
   else
      Print("Indicateur "+shortName+" n'a pas trouvé. window = "+(string)window);
  }
```

Voir aussi

ObjectCreate(), ObjectFind()

© 2000-2015, MetaQuotes Software Corp.

ChartTimePriceToXY

Convertit les coordonnées du graphique de la représentation temps/prix pour les coordonnées selon axe X et Y.

```
bool  ChartTimePriceToXY(
   long           chart_id,      // l'identificateur du graphique
   int            sub_window,    //le numéro de la sous-fenêtre
   datetime       time,          // le temps sur le graphique
   double         price,         // le temps sur le graphique
   int&           x,             // la coordonnée X pour le temps sur le graphique
   int&           y              // la coordonnée Y pour le prix sur le graphique
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

time

[in] La signification du temps sur le graphique, pour laquelle sera reçue la valeur en pixels sur l'axe X. Le début des coordonnées se trouve dans un angle gauche supérieur de la fenêtre principale du graphique.

price

[in] La signification du prix sur le graphique, pour laquelle sera reçue la valeur en pixels sur l'axe Y. Le début des coordonnées se trouve dans un angle gauche supérieur de la fenêtre principale du graphique.

x

[out] La variable, où sera reçue la transformation du temps en coordonnée X.

y

[out] La variable, où sera reçue la transformation du prix en coordonnée Y.

La valeur rendue

Rend true en cas de l'exécution réussie, autrement - false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError () .

Voir aussi

ChartXYToTimePrice()

© 2000-2015, MetaQuotes Software Corp.

ChartXYToTimePrice

Convertit les coordonnées X et Y du graphique en valeurs le temps et le prix.

```
bool  ChartXYToTimePrice(
   long           chart_id,      // l'identificateur du graphique
   int            x,             // la coordonnée X sur le graphique
   int            y,             // la coordonnée Y sur le graphique
   int&           sub_window,    // le numéro de la sous-fenêtre
   datetime&      time,          // le temps sur le graphique
   double&        price          // le prix sur le graphique
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

x

[in] La coordonnée X.

y

[in] La coordonnée Y.

sub_window

[out] La variable, où sera inscrit le numéro de la sous- fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

time

[out] La signification du temps sur le graphique, pour laquelle sera reçue la valeur en pixels sur l'axe X. Le début des coordonnées se trouve dans un angle gauche supérieur de la fenêtre principale du graphique.

price

[out] La signification du prix sur le graphique, pour laquelle sera reçue la valeur en pixels sur l'axe Y. Le début des coordonnées se trouve dans un angle gauche supérieur de la fenêtre principale du graphique.

La valeur rendue

Rend true en cas de l'exécution réussie, autrement - false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError () .

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- déduisons les paramètres de l'événement sur le graphique
   Comment(__FUNCTION__,": id=",id," lparam=",lparam," dparam=",dparam," sparam=",spar
//--- Si ce sont les événements du clic de la souris sur le graphique
   if(id==CHARTEVENT_CLICK)
     {
      //--- préparons les variables
      int      x     =(int)lparam;
      int      y     =(int)dparam;
      datetime dt    =0;
      double   price =0;
      int      window=0;
      //--- transformons les coordonnées X et Y dans les termes la date/temps
      if(ChartXYToTimePrice(0,x,y,window,dt,price))
        {
         PrintFormat("Window=%d X=%d  Y=%d  =>  Time=%s  Price=%G",window,x,y,TimeToSt
         //--- faisons la transformation inverse: (X,Y) => (Time,Price)
         if(ChartTimePriceToXY(0,window,dt,price,x,y))
            PrintFormat("Time=%s  Price=%G  =>  X=%d  Y=%d",TimeToString(dt),price,x,y
         else
            Print("ChartTimePriceToXY return error code: ",GetLastError());
         //--- delete lines
         ObjectDelete(0,"V Line");
         ObjectDelete(0,"H Line");
         //--- create horizontal and vertical lines of the crosshair
         ObjectCreate(0,"H Line",OBJ_HLINE,window,dt,price);
         ObjectCreate(0,"V Line",OBJ_VLINE,window,dt,price);
         ChartRedraw(0);
        }
      else
         Print("ChartXYToTimePrice return error code: ",GetLastError());
      Print("+------------------------------------------------------------------+");
     }
  }
```

Voir aussi

ChartTimePriceToXY()

© 2000-2015, MetaQuotes Software Corp.

ChartOpen

Ouvre un nouveau graphique avec le symbole indiqué et la période.

```
long  ChartOpen(
   string           symbol,     // nom du symbole
   ENUM_TIMEFRAMES  period      // période
   );
```

Paramètres

symbol

[in] Le symbole du graphique. NULL signifit le symbole du graphique courant (auquel l'expert est attaché).

period

[in] La période du graphique (le temps trame). Peut prendre une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 est la période du graphique courant.

La valeur rendue

A l'ouverture réussie du graphique la fonction rend l'identificateur du graphique. Autrement rend 0.

Note

Le nombre possible maximum de graphiques simultanément ouverts dans le terminal ne peut pas ne peut pas excéder la valeur CHARTS_MAX.

© 2000-2015, MetaQuotes Software Corp.

ChartFirst

Rend l'identificateur du premier graphique du terminal de client.

```
long  ChartFirst();
```

La valeur rendue

L'identificateur du graphique.

© 2000-2015, MetaQuotes Software Corp.

ChartNext

La fonction rend l'identificateur du graphique, suivant de l'indiqué.

```
long  ChartNext(
   long  chart_id      // identificateur du graphique
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 ne signifie pas le graphique courant. 0 signifie "rendre l'identificateur du premier graphique".

La valeur rendue

L'identificateur du graphique. Si la liste des graphiques s'est achevé, la fonction rend -1.

Exemple:

```
//--- variables pour les identificateurs des graphiques
   long currChart,prevChart=ChartFirst();
   int i=0,limit=100;
   Print("ChartFirst = ",ChartSymbol(prevChart)," ID = ",prevChart);
   while(i<limit)// nous avons certainement pas plus de 100 graphiques ouverts
     {
      currChart=ChartNext(prevChart); // nous recevons un nouveau graphique à la base
      if(currChart<0) break;          // ont atteint la fin de la liste des graphiques
      Print(i,ChartSymbol(currChart)," ID = ",currChart);
      prevChart=currChart;// sauvegardons  l'identificateur du graphique courant pour
      i++;// n'oublions pas d'augmenter le compteur
     }
```

© 2000-2015, MetaQuotes Software Corp.

ChartClose

Ferme le graphique indiqué.

```
bool  ChartClose(
   long  chart_id=0      // identificateur du graphique
   );
```

Paramètres

chart_id=0

[in] L'identificateur du graphique. 0 signifit le graphique courant.

La valeur rendue

Rend true en cas du succès, autrement rend false.

© 2000-2015, MetaQuotes Software Corp.

ChartSymbol

Rend le nom du symbole du graphique indiqué.

```
string  ChartSymbol(
   long  chart_id=0      // identificateur du graphique
   );
```

Paramètres

chart_id=0

[in] L'identificateur du graphique. 0 signifit le graphique courant.

La valeur rendue

Si le graphique n'existe pas, la chaîne vide se rend.

Voir aussi

ChartSetSymbolPeriod

© 2000-2015, MetaQuotes Software Corp.

ChartPeriod

Rend la valeur de la période di graphique indiqué.

```
ENUM_TIMEFRAMES  ChartPeriod(
   long  chart_id=0      // identificateur du graphique
   );
```

Paramètres

chart_id=0

[in] L'identificateur du graphique. 0 signifit le graphique courant.

La valeur rendue

La valeur du type ENUM_TIMEFRAMES. Si le graphique n'existe pas, revient 0.

© 2000-2015, MetaQuotes Software Corp.

ChartRedraw

Appelle la copie forcée du graphique indiqué.

```
void  ChartRedraw(
   long  chart_id=0      // identificateur du graphique
   );
```

Paramètres

chart_id=0

[in] L'identificateur du graphique. 0 signifit le graphique courant.

Note

Est appliqué d'habitude après le changement des propriétés des objets.

Voir aussi

Les objets graphiques

© 2000-2015, MetaQuotes Software Corp.

ChartSetDouble

Специфие la valeur de la propriété correspondante du graphique indiqué. La propriété du graphique doit être du type double. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartSetDouble(
   long     chart_id,      // identificateur du graphique
   int      prop_id,       // identificateur de la propriété
   double   value          // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être une des valeurs de l'énumération ENUM_CHART_PROPERTY_DOUBLE (sauf les propriétés read-only).

value

[in] La valeur de la propriété.

La valeur rendue

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

ChartSetInteger

Spécifie la valeur de la propriété correspondante du graphique indiqué. La propriété du graphique doit être des types datetime, int, color, bool ou char. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartSetInteger(
   long    chart_id,      // identificateur du graphique
   int     prop_id,       // identificateur de la propriété
   long    value          // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être une des valeurs de l'énumération ENUM_CHART_PROPERTY_INTEGER (sauf les propriétés read-only).

value

[in] La valeur de la propriété.

La valeur rendue

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

ChartSetString

Специфие la valeur de la propriété correspondante du graphique indiqué. La propriété du graphique doit être du type string. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartSetString(
   long  chart_id,         // identificateur du graphique
   int   prop_id,          // identificateur de la propriété
   string   str_value      // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être une des valeurs de l'énumération ENUM_CHART_PROPERTY_STRING (sauf les propriétés read-only).

str_value

[in] La chaîne pour l'installation de la propriété. La longueur de la chaîne ne peut pas excéder 2045 caractères (les caractères superflus seront coupés).

La valeur rendue

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError().

Note

La fonction ChartSetString peut être utilisée pour la sortie des commentaires sur le graphique au lieu de la fonction Comment.

Exemple:

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
   string comment=StringFormat("Sortons les prix:\nAsk = %G\nBid = %G\nSpread = %d",
                               Ask,Bid,Spread);
   ChartSetString(0,CHART_COMMENT,comment);
  }
```

Voir aussi

Comment, ChartGetString

© 2000-2015, MetaQuotes Software Corp.

ChartGetDouble

Rend la valeur de la propriété correspondante du graphique indiqué. La propriété du graphique doit être du type double. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
double  ChartGetDouble(
   long  chart_id,          // identificateur du graphique
   int   prop_id,           // identificateur de la propriété
   int   sub_window=0       // numéro de la sous-fenêtre, si c'est nécessaire
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  ChartGetDouble(
   long    chart_id,        // identificateur du graphique
   int     prop_id,         // identificateur de la propriété
   int     sub_window,      // numéro de la sous-fenêtre
   double& double_var       // acceptons ici la valeur de la propriété
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être une des valeurs de l'énumération ENUM_CHART_PROPERTY_DOUBLE.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. Pour la première variante la valeur est égale à 0 par défaut (une fenêtre principale du graphique). La plupart des propriétés ne demandent pas l'indication du numéro de la sous-fenêtre.

double_var

[out] La variable du type double, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type double.

Pour la deuxième variante de l'appel rend true, si la propriété donnée est supportée et la valeur était placé à la variable double_var, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d' appeler la fonction GetLastError().

Exemple:

```
void OnStart()
  {
   double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,0);
```

© 2000-2015, MetaQuotes Software Corp.

```
   double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,0);
   Print("CHART_PRICE_MIN = ",priceMin);
   Print("CHART_PRICE_MAX = ",priceMax);
  }
```

© 2000-2015, MetaQuotes Software Corp.

ChartGetInteger

Rend la valeur de la propriété correspondante du graphique indiqué. La propriété du graphique doit être des types datetime, int ou bool. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
long  ChartGetInteger(
   long  chart_id,          // identificateur du graphique
   int   prop_id,           // identificateur de la propriété
   int   sub_window=0       // numéro de la sous-fenêtre, si c'est nécessaire
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  ChartGetInteger(
   long    chart_id,        // identificateur du graphique
   int     prop_id,         // identificateur de la propriété
   int     sub_window,      // numéro de la sous-fenêtre
   long&   long_var         // acceptons ici la valeur de la propriété
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être une des valeurs de l'énumération ENUM_CHART_PROPERTY_INTEGER.

sub_window

[in] Pour la première variante la valeur est égale à 0 par défaut (une fenêtre principale du graphique). La plupart des propriétés ne demandent pas l'indication du numéro de la sous-fenêtre.

long_var

[out] La valeur du type long, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type long.

Pour la deuxième variante de l'appel rend true, si la propriété donnée est supportée et la valeur était placé à la variable long_var, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d' appeler la fonction GetLastError().

Exemple:

```
void OnStart()
  {
   int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
```

© 2000-2015, MetaQuotes Software Corp.

```
   Print("CHART_HEIGHT_IN_PIXELS = ",height," pixels");
   Print("CHART_WIDTH_IN_PIXELS = ",width," pixels");
  }
```

© 2000-2015, MetaQuotes Software Corp.

ChartGetString

Rend la valeur de la propriété correspondante du graphique indiqué. La propriété du graphique doit être du type string. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
string  ChartGetString(
   long   chart_id,          // identificateur du graphique
   int    prop_id            // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  ChartGetString(
   long     chart_id,        // identificateur du graphique
   int      prop_id,         // identificateur de la propriété
   string&  string_var       // acceptons ici la valeur de la propriété
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

prop_id

[in] L'identificateur de la propriété du graphique. La valeur peut être une des valeurs de l'énumération ENUM_CHART_PROPERTY_STRING.

string_var

[out] La variable du type string, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type string.

Pour la deuxième variante de l'appel rend true, si la propriété donnée est supportée et la valeur était placé à la variable string_var, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d' appeler la fonction GetLastError().

Note

La fonction ChartGetString peut être utilisée pour la lecture des commentaires affichés sur le graphique par les fonctions Comment ou ChartSetString.

Exemple:

```
void OnStart()
  {
   ChartSetString(0,CHART_COMMENT,"Test comment.\nSecond line.\nThird!");
   ChartRedraw();
   Sleep(1000);
   string comm=ChartGetString(0,CHART_COMMENT);
```

© 2000-2015, MetaQuotes Software Corp.

```
   Print(comm);
  }
```

Voir aussi

Comment, ChartSetString

© 2000-2015, MetaQuotes Software Corp.

ChartNavigate

Réalise le décalage du graphique indiqué au nombre indiqué de barres par rapport à la position indiquée du graphique.

```
bool  ChartNavigate(
   long  chart_id,      // identificateur du graphique
   int   position,      // position
   int   shift=0        // valeur du décalage
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

position

[in] La position de graphique pour exécuter le décalage. La valeur peut être une des valeurs de l'énumération ENUM_CHART_POSITION.

shift=0

[in] Le nombre de barres, auxquels il faut déplacer le graphique. La valeur positive signifie le décalage à droite (à la fin du graphique), la valeur négative signifie le décalage à gauche (vers le début du graphique). Le décalage nul est justifié, quand on produit la navigation vers le début ou la fin du graphique.

La valeur rendue

Rend true en cas du succès, autrement rend false.

Exemple:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- recevons le handle de l'indicateur actuel
   long handle=ChartID();
   string comm="";
   if(handle>0) // en cas de succès, configurons supplémentairement
     {
      //--- désactivons l'autodéfilement
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- établissons l'alinéa du bord droit du graphique
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- affichons en forme des bougies
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- établir le mode de l'affichage des volumes de tick
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- préparons le texte pour la sortie dans Comment()
   comm="Faisons défiler 10 barres à droite du début de l'histore";
   //--- déduisons le commentaire
   Comment(comm);
   //--- faisons défiler 10 barres à droite du début de l'histore
   ChartNavigate(handle,CHART_BEGIN,10);
   //--- recevons le numéro de la première barre affichée sur le graphique (le numé
   long first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   //--- ajoutons le symbole du transfert de la ligne
   comm=comm+"\r\n";
   //--- completons le commentaire
   comm=comm+"La première barre sur le graphique a le numéro"+IntegerToString(first
   //--- déduisons le commentaire
   Comment(comm);
   //--- attendons 5 secondes pour avoir le temps de voir comment avance le graphic
   Sleep(5000);

   //--- écrivons le texte du commentaire
   comm=comm+"\r\n"+"Faisons défiler 10 barres à gauche du bord droit du graphique"
   Comment(comm);
   //--- faisons défiler 10 barres à gauche du bord droit du graphique
   ChartNavigate(handle,CHART_END,-10);
   //--- recevons le numéro de la première barre affichée sur le graphique (le numé
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
   comm=comm+"La première barre sur le graphique a le numéro"+IntegerToString(first
   Comment(comm);
   //--- attendons 5 secondes pour avoir le temps de voir comment avance le graphic
   Sleep(5000);

   //--- un nouveau bloc du défilement du graphique
   comm=comm+"\r\n"+"Faisons défiler 300 barres à droite du début de l'histore";
   Comment(comm);
   //--- faisons défiler 300 barres à droite du début de l'histore
   ChartNavigate(handle,CHART_BEGIN,300);
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
   comm=comm+"La première barre sur le graphique a le numéro"+IntegerToString(first
   Comment(comm);
   //--- attendons 5 secondes pour avoir le temps de voir comment avance le graphic
   Sleep(5000);

   //--- un nouveau bloc du défilement du graphique
   comm=comm+"\r\n"+"Faisons défiler 300 barres à gauche du bord droit du graphique
   Comment(comm);
   //--- faisons défiler 300 barres à gauche du bord droit du graphique
   ChartNavigate(handle,CHART_END,-300);
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
```

© 2000-2015, MetaQuotes Software Corp.

```
      comm=comm+"La première barre sur le graphique a le numéro"+IntegerToString(first
      Comment(comm);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

ChartID

Rend l'identificateur du graphique courant.

```
long  ChartID();
```

La valeur rendue

La valeur du type long.

© 2000-2015, MetaQuotes Software Corp.

ChartIndicatorAdd

Ajoute l'indicateur avec le handle indiqué sur la fenêtre indiquée du graphique. L'indicateur et le graphique doivent être construits sur le même symbole et le temps trame.

```
bool  ChartIndicatorAdd(
   long  chart_id,                  // l'identificateur du graphique
   int   sub_window                 // le numéro de la sous-fenêtre
   int   indicator_handle           //le handle indiqué
   );
```

Les paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie une fenêtre principale du graphique. Pour ajouter un indicateur dans une nouvelle fenêtre, le paramètre doit être plus grand que l'index de la dernière fenêtre existante, alors il doit être égal à CHART_WINDOWS_TOTAL. Si la valeur du paramètre dépasse la valeur CHART_WINDOWS_TOTAL, alors la nouvelle fenêtre ne sera pas créée et l'indicateur ne sera pas ajouté.

indicator_handle

[in] Le handle de l'indicateur.

La valeur rendue

Rend true en cas du succès, autrement rend false. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError(). L'erreur 4114 signifie que le graphique et l'indicateur ajouté se distinguent selon le symbole ou le temps trame.

Note

Si sur une principale fenêtre du graphique on ajoute l'indicateur, qui doit être dessiné dans le sous-fenêtre séparé (par exemple, iMACD inséré ou l'indicateur d'utilisateur avec la propriété indiquée #property indicator_separate_window), un tel indicateur peut être invisible, bien qu'il sera dans la liste des indicateurs. Cela signifie que l'échelle de l'indicateur donné se distingue de l'échelle du graphique des prix et les valeurs de l'indicateur ajouté ne se sont pas introduits dans la gamme affichée du graphique de prix. Dans ce cas GetLastError() rendra le code nul signifiant l'absence de l'erreur. On pourra observer les valeurs d'un tel indicateur "invisible" dans "Fenêtre des données" et recevoir des autres programmes MQL5.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
#property description "L'expert pour la démonstration du travail avec la fonction Char
#property description "Après le lancement sur le graphique (et la réception de l'erreu
#property description "les propriétés du conseiller et spécifiez les paramètres justes
#property description "L'indicateur MACD sera ajouté sur le graphique."

//--- input parameters
input string          symbol="AUDUSD";     // le nom du symbole
input ENUM_TIMEFRAMES period=PERIOD_M12;   // le temps trame
input int    fast_ema_period=12;           // la période de la moyenne rapide MACD
input int    slow_ema_period=26;           // la période de la moyenne lente MACD
input int      signal_period=9;            // la période de la prise de moyenne de la d
input ENUM_APPLIED_PRICE apr=PRICE_CLOSE; // le type du prix pour le calcul MACD

int indicator_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_period,
//--- essayons ajouter l'indicateur sur le graphique
   if(!AddIndicator())
     {
      //--- la fonction AddIndicator () a refusé d'ajouter l'indicateur sur le graphic
      int answer=MessageBox("Quand même, essayer d'ajouter MACD sur le graphique?",
                            "Le symbole et/ou le temps trame sont spécifiés incorrecte
                            MB_YESNO // les boutons du choix "Yes" et "No" seront mont
                            );
      //--- si l'utilisateur insiste en tout cas sur l'utilisation incorrecte de Chart
      if(answer==IDYES)
        {
         //--- d'abord communiquons sur cela au journal
         PrintFormat("L'attention! %s: Essayons ajouter l'indicateur MACD(%s/%s) sur l
                     __FUNCTION__,symbol,EnumToString(period),_Symbol,EnumToString(_Pe
         //--- recevons le numéro du nouveau sous-fenêtre, où tentons d'ajouter l'indi
         int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
         //--- maintenant faisons la tentative condamnée à l'erreur
         if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
            PrintFormat("On n'a pas réussi à ajouter l'indicateur MACD sur la fenêtre
                        subwindow,GetLastError());
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
// l'expert ne fait rien
  }
//+------------------------------------------------------------------+
//| La fonction de la vérification et le supplément de l'indicateur sur le graphique
//+------------------------------------------------------------------+
bool AddIndicator()
  {
//--- le message déduit
   string message;
//--- vérifions sur la coïncidence le symbole de l'indicateur et le symbole du graphic
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(symbol!=_Symbol)
     {
      message="La démonstration de l'utilisation de la fonction Demo_ChartIndicatorAdd
      message=message+"\r\n";
      message=message+"On ne peut pas ajouter sur le graphique l'indicateur calculé su
      message=message+"\r\n";
      message=message+"Indiquez le symbole du graphique dans les propriétés de l'exper
      Alert(message);
      //--- la sortie anticipé, n'ajoutons pas l'indicateur sur le graphique
      return false;
     }
//--- vérifions sur la coïncidence le temps trame de l'indicateur et le temps trame du
   if(period!=_Period)
     {
      message="On ne peut pas ajouter sur le graphique l'indicateur, calculé sur un au
      message=message+"\r\n";
      message=message+"Indiquez dans les propriétés de l'expert le temps trame, qui es
      Alert(message);
      //--- la sortie anticipé, n'ajoutons pas l'indicateur sur le graphique
      return false;
     }
//--- toutes les vérificatios ont passé, le symbole et la période de l'indicateur corr
   if(indicator_handle==INVALID_HANDLE)
     {
      Print(__FUNCTION__," Créons l'indicateur MACD");
      indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_peri
      if(indicator_handle==INVALID_HANDLE)
        {
         Print("On n'a pas réussi à créer l'indicateur MACD. Le code de l'erreur ",Get
        }
     }
//--- obliterons le code de l'erreur
   ResetLastError();
//---appliquons l'indicateur au graphique
   Print(__FUNCTION__," Ajoutons l'indicateur MACD sur le graphique");
   Print("MACD est construit sur ",symbol,"/",EnumToString(period));
//--- recevons le numéro du nouveau sous-fenêtre, où ajoutons l'indicateur MACD
   int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   PrintFormat("Ajoutons l'indicateur MACD ) à la fenêtre du graphique %d ",subwindow)
   if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
     {
      PrintFormat("On n'a pas réussi à ajouter l'indicateur MACD sur la fenêtre du gra
                  subwindow,GetLastError());
     }
//--- le supplément de l'indicateur sur le graphique a passé avec succès
   return(true);
  }
```

Voir aussi

ChartIndicatorDelete(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate()

© 2000-2015, MetaQuotes Software Corp.

ChartIndicatorDelete

Supprime l'indicateur avec le nom spécifié de la fenêtre indiquée du graphique.

```
bool  ChartIndicatorDelete(
   long           chart_id,              // l'identificateur du graphique
   int            sub_window             // le numéro de la sous-fenêtre
   const string   indicator_shortname    // le nom court de l'indicateur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

const indicator_shortname

[in] Le nom court de l'indicateur, qui est spécifié dans la propriété INDICATOR_SHORTNAME par la fonction IndicatorSetString(). On peut recevoir le nom court de l'indicateur par la fonction ChartIndicatorName().

La valeur rendue

Rend true si l'indicateur a été supprimé avec succès, autrement rend false. Pour recevoir l'information sur l'erreur il faut appeler la fonction GetLastError () .

Note

S'il y a plusieurs indicateurs avec le même nom court dans la sous-fenêtre spécifiée du graphique, sera supprimé le premier indicateur.

Si les autres indicateurs sont construits sur les valeurs de l'indicateur supprimé sur le même graphique, ils seront également supprimés.

Il ne faut pas confondre un nom court et de l'indicateur et le nom du fichier qui est spécifié pendant la création de l'indicateur par les fonctions iCustom() et IndicatorCreate(). Si le nom court de l'indicateur n'est pas spécifié explicitement, alors pendant la compilation le nom du fichier avec le code initial de l'indicateur y est spécifié.

La suppression de l'indicateur du graphique ne signifie pas que la partie calculé de l'indicateur sera aussi supprimée de la mémoire du terminal. Pour vider le handle de l'indicateur utiliser la fonction IndicatorRelease().

Il est nécessaire de former correctement un nom court de l'indicateur, qui s'écrit dans la propriété INDICATOR_SHORTNAME à l'aide de la fonction IndicatorSetString () . Nous recommandons que le nom court contienne les valeurs des paramètres d'entrée de l'indicateur, puisque l'identification de l'indicateur supprimé du graphique dans la fonction ChartIndicatorDelete() est faite selon le nom court.

L'exemple de la suppression de l'indicateur pendant la mauvaise initialisation:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                    Demo_ChartIndicatorDelete.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      first_param=1;
input int      second_param=2;
input int      third_param=3;
input bool     wrong_init=true;
//--- indicator buffers
double         HistogramBuffer[];
string         shortname;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int res=INIT_SUCCEEDED;
//--- attachons le tableau HistogramBuffer au tampon d'indicateur
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- construisons le nom court de l'indicateur à la base des paramètres d'entrée
   shortname=StringFormat("Demo_ChartIndicatorDelete(%d,%d,%d)",
                          first_param,second_param,third_param);
   IndicatorSetString(INDICATOR_SHORTNAME,shortname);
//--- si on a spécifié l'achèvement forcé de l'indicateur, rendons la valeur non nulle
   if(wrong_init) res=INIT_FAILED;
   return(res);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- la position initiale pour le travail dans la boucle
   int start=prev_calculated-1;
   if(start<0) start=0;
//--- remplissons le tampon d'indicateur par les valeurs
   for(int i=start;i<rates_total;i++)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      HistogramBuffer[i]=close[i];
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| gestionnaire d'événements Deinit                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   PrintFormat("%s: le code de la raison de la désinitialisation =%d",__FUNCTION__,rea
   if(reason==REASON_INITFAILED)
     {
      PrintFormat("L'indicateur avec le nom court %s (le fichier %s) se supprime du gr
      int window=ChartWindowFind();
      bool res=ChartIndicatorDelete(0,window,shortname);
      //--- analysons le résultat de l'appel ChartIndicatorDelete()
      if(!res)
        {
         PrintFormat("On n'a pas réussi à supprimer l'indicateur %s de la fenêtre #%d.
                     shortname,window,GetLastError());
        }
     }
  }
```

Voir aussi

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2015, MetaQuotes Software Corp.

ChartIndicatorGet

Rend le handle de l'indicateur avec le nom court indiqué sur la fenêtre indiquée du graphique.

```
int  ChartIndicatorGet(
   long           chart_id,              // l'identificateur du graphique
   int            sub_window             // le numéro de la sous-fenêtre
   const string   indicator_shortname    // le nom court de l'indicateur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

const indicator_shortname

[in] Le nom court de l'indicateur, qui est spécifié dans la propriété INDICATOR_SHORTNAME par la fonction IndicatorSetString(). On peut recevoir le nom court de l'indicateur par la fonction ChartIndicatorName().

La valeur rendue

Rend le handle de l'indicateur en cas de l'exécution réussie, autrement INVALID_HANDLE. Pour recevoir l'informationl'erreur, il faut appeler la fonctionGetLastError().

Note

Il est nécessaire de former correctement un nom court de l'indicateur à sa création, qui à l'aide de la fonction IndicatorSetString() est enregistré à la propriétéINDICATOR_SHORTNAME. Nous recommandons que le nom court contienne les valeurs des paramètres d'entrée de l'indicateur, puisque l'identification de l'indicateur dans la fonction ChartIndicatorGet() est faite selon le nom court.

Un autre moyen de l'identification de l'indicateur – recevoir la liste de ses paramètres selon le handle donné à l'aide de la fonction IndicatorParameters() et puis analyser les valeurs reçues.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- le nombre de fenêtres sur le graphique (il y a toujours au moins une fenêtre
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   //--- passons par les fenêtres
   for(int w=0;w<windows;w++)
     {
      //--- combien d'indicateurs dans cette fenêtre/sous-fenêtre
      int total=ChartIndicatorsTotal(0,w);
      //---trions tous les indicateurs dans la fenêtre
      for(int i=0;i<total;i++)
        {
         //--- recevons le nom court de l'indicateur
         string name=ChartIndicatorName(0,w,i);
         //--- recevons le handle de l'indicateur
         int handle=ChartIndicatorGet(0,w,name);
         //--- déduisons dans un journal
         PrintFormat("Window=%d,  index=%d,  name=%s,  handle=%d",w,i,name,handle);
         //--- обязательно освобождаем хендл индикатора, как только он становится не н
         IndicatorRelease(handle);
        }
     }
  }
```

Voir aussi

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), IndicatorParameters()

© 2000-2015, MetaQuotes Software Corp.

ChartIndicatorName

Rend le nom court de l'indicateur selon le numéro dans la liste des indicateurs sur la fenêtre indiquée du graphique.

```
string  ChartIndicatorName(
   long   chart_id,       // l'identificateur du graphique
   int    sub_window      // le numéro de la sous-fenêtre
   int    index           // L'index de l'indicateur dans la liste des indicateurs ajout
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

index

[in] L'index d'un indicateur dans la liste des indicateurs. Le numérotage des indicateurs commence par le zéro, c'est-à-dire le premier indicateur dans la liste a l'index nul. On peut recevoir le nombre d'indicateurs dans la liste par la fonctionChartIndicatorsTotal().

La valeur rendue

Le nom court de l'indicateur, qui est spécifié dans la propriété INDICATOR_SHORTNAMEpar la fonctionIndicatorSetString(). On peut recevoir le nom court de l'indicateur par la fonctionChartIndicatorName(). Pour recevoir l'information surl'erreur, il faut appeler la fonction GetLastError().

Note

Il ne faut pas confondre un nom court et de l'indicateur et le nom du fichier qui est spécifié pendant la création de l'indicateur par les fonctionsiCustom() et IndicatorCreate(). Si le nom court de l'indicateur n'est pas spécifié explicitement, alors pendant la compilation le nom du fichier avec le code initial de l'indicateur y est spécifié.

La suppression de l'indicateur du graphique ne signifie pas que la partie calculé de l'indicateur sera aussi supprimée de la mémoire du terminal. Pour vider le handle de l'indicateur utiliser la fonctionIndicatorRelease().

Il est nécessaire de former correctement un nom court de l'indicateur, qui s'écrit dans la propriété INDICATOR_SHORTNAME à l'aide de la fonctionIndicatorSetString () . Nous recommandons que le nom court contienne les valeurs des paramètres d'entrée de l'indicateur, puisque l'identification de l'indicateur supprimé du graphique dans la fonctionChartIndicatorDelete()est faite selon le nom court.

Voir aussi

ChartIndicatorAdd(), ChartIndicatorDelete(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2015, MetaQuotes Software Corp.

ChartIndicatorsTotal

Rend le nombre de tous indicateurs qui sont attachés à la fenêtre indiquée du graphique.

```
int  ChartIndicatorsTotal(
   long  chart_id,      // l'identificateur du graphique
   int   sub_window     // le numéro de la sous-fenêtre
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique actuel.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre principale du graphique.

La valeur rendue

Le nombre d'indicateurs sur la fenêtre spécifiée du graphique. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError ().

Note

La fonction est conçue pour trier tous les indicateurs attachés à ce graphique. On peut recevoir le nombre de toutes les fenêtres du graphique de la propriétéCHART_WINDOWS_TOTAL par la fonctionChartGetInteger().

Voir aussi

ChartIndicatorAdd(), ChartIndicatorDelete(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2015, MetaQuotes Software Corp.

ChartWindowOnDropped

Rend le numéro de sous-fenêtre du graphique, sur lequel on jette par la souris l'expert donné, le script, l'objet ou l'indicateur. 0 signifie une fenêtre principale du graphique.

```
int  ChartWindowOnDropped();
```

La valeur rendue

La valeur du type int.

Exemple:

```
   int myWindow=ChartWindowOnDropped();
   int windowsTotal=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   Print("Script est lancé sur la fenêtre #"+myWindow+
         ". Au total les fenêtres sur le graphique "+ChartSymbol()+": ",windowsTotal);
```

Voir aussi

ChartPriceOnDropped, ChartTimeOnDropped, ChartXOnDropped, ChartYOnDropped

© 2000-2015, MetaQuotes Software Corp.

ChartPriceOnDropped

Rend la coordonnée de prix correspondant au point, dans laquelle on jette par la souris l'expert donné ou le script.

```
double  ChartPriceOnDropped();
```

La valeur rendue

La valeur du type double.

Exemple:

```
   double p=ChartPriceOnDropped();
   Print("ChartPriceOnDropped() = ",p);
```

Voir aussi

ChartXOnDropped, ChartYOnDropped

© 2000-2015, MetaQuotes Software Corp.

ChartTimeOnDropped

Rend la coordonnée temporaire correspondant au point, dans laquelle on jette par la souris l'expert donné ou le script.

```
datetime  ChartTimeOnDropped();
```

La valeur rendue

La valeur du type datetime.

Exemple:

```
   datetime t=ChartTimeOnDropped();
   Print("Script wasdropped on the "+t);
```

Voir aussi

ChartXOnDropped, ChartYOnDropped

© 2000-2015, MetaQuotes Software Corp.

ChartXOnDropped

Rend la coordonnée selon l'axe X, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris.

```
int  ChartXOnDropped();
```

La valeur rendue

La valeur de la coordonnée X.

Note

L'axe X est dirigé de gauche à droite.

Exemple:

```
   int X=ChartXOnDropped();
   int Y=ChartYOnDropped();
   Print("(X,Y) = ("+X+","+Y+")");
```

Voir aussi

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2015, MetaQuotes Software Corp.

ChartYOnDropped

Rend la coordonnée selon l'axe Y, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris.

```
int  ChartYOnDropped();
```

La valeur rendue

La valeur de la coordonnée Y.

Note

L'axe Y est dirigé de haut en bas.

Voir aussi

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2015, MetaQuotes Software Corp.

ChartSetSymbolPeriod

Change les valeurs du symbole et la période du graphique indiqué. La fonction travaille asynchronement, c'est-à-dire rend la commande et n'attend pas la fin de son exécution. Отданная команда поступает в очередь сообщений графика и выполняется только после обработки всех предыдущих команд.

```
bool  ChartSetSymbolPeriod(
   long             chart_id,      // identificateur du graphique
   string           symbol,        // nom du symbole
   ENUM_TIMEFRAMES  period         // période
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

symbol

[in] Le symbole du graphique. NULL signifit le symbole du graphique courant (auquel l'expert est attaché)

period

[in] La période du graphique (le temps trame). Peut prendre une des valeurs de l'énumération ENUM_TIMEFRAMES. 0 est la période du graphique courant.

La valeur rendue

Возвращает true в случае удачного помещения команды в очередь графика, иначе false. Pour recevoir l'information sur l'erreuril faut appeler la fonction GetLastError().

Note

Le remplacement du symbole/la période entraîne la réinitialisation d'expert, attaché au graphique correspondant.

Voir aussi

ChartSymbol, ChartPeriod

© 2000-2015, MetaQuotes Software Corp.

ChartScreenShot

La fonction assure le screenshot du graphique indiqué dans son état courant dans le format GIF, PNG ou BMP en fonction de l'extension spécifiée.

```
bool  ChartScreenShot(
   long             chart_id,                  // identificateur du graphique
   string           filename,                  // nom du fichier
   int              width,                     // largeur
   int              height,                    // hauteur
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // type de l'alignement
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifit le graphique courant.

filename

[in] Le nom du fichier de screenshot. Ne peut pas excéder 63 caractères. Le screenshot se place au répertoire \Files.

width

[in] La largeur de screenshot en pixels

height

[in] La hauteur de screenshot en pixels

align_mode=ALIGN_RIGHT

[in] Le mode de la sortie de screenshot étroit. les valeurs de l'énumération ENUM_ALIGN_MODE. ALIGN_RIGHT signifie l'alignement selon la marge droite (la sortie de la fin). ALIGN_LEFT signifie l'alignement selon une marge gauche.

La valeur rendue

Rend true en cas du succès, autrement rend false.

Note

S'il faut de faire un screenshot d'un graphique d'une certaine position, tout d'abord il est nécessaire de positionner le graphique en utilisant la fonction ChartNavigate(). Si la dimension horizontale du screenshot est plus petite que la fenêtre de graphique, la partie droite de la fenêtre du graphique, ou une gauche partie sont affichées, en fonction de la valeur du paramètre align_mode.

Exemple:

```
#property description "Le conseiller présente la création de la série des captures d'é
#property description "à l'aide de la fonction ChartScreenShot(). Le nom du fichier"
#property description "est déduit aussi sur le graphique pour la commodité. La hauteu

#define        WIDTH  800     // la largeur du dessin pour l'appel de ChartScreenShot
#define        HEIGHT 600     // La hauteur du dessin pour l'appel de ChartScreenShot
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- input parameters
input int      pictures=5;    // la quantité de dessins dans une série
int            mode=-1;       // -1 signifie le décalage vers le bord droit du graphic
int            bars_shift=300;// le nombre de barres au défilement du graphique par la
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//---désactivons l'autodéfilement du graphique
   ChartSetInteger(0,CHART_AUTOSCROLL,false);
//--- établissons l'alinéa du bord droit du graphique
   ChartSetInteger(0,CHART_SHIFT,true);
//---établissons l'affichage du graphique en forme des bougies
   ChartSetInteger(0,CHART_MODE,CHART_CANDLES);
//---
   Print("La préparation du conseiller au travail est terminée");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- la sortie du nom de la fonction, du temps de l'appel et de l'identificateur de l
   Print(__FUNCTION__,TimeCurrent(),"   id=",id,"   mode=",mode);
//--- le traitement de l'événement CHARTEVENT_CLICK ("La pression par la souris sur le
   if(id==CHARTEVENT_CLICK)
     {
      //--- le déplacement initial du bord du graphique
      int pos=0;
      //--- le mode du travail avec le bord gauche du graphique
      if(mode>0)
        {
         //--- défilons le graphique vers le bord gauche
         ChartNavigate(0,CHART_BEGIN,pos);
         for(int i=0;i<pictures;i++)
           {
            //---préparons la signature sur le graphique et le nom pour le fichier
```

© 2000-2015, MetaQuotes Software Corp.

```
            string name="ChartScreenShot"+"CHART_BEGIN"+string(pos)+".gif";
            //--- la conclusion du nom sur le graphique en forme du commentaire
            Comment(name);
            //--- sauvegardons la capture d'écran au dossier le répertoire_du terminal
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_LEFT))
               Print("Nous avons sauvegardé la capture d'écran ",name);
            //---
            pos+=bars_shift;
            //---donnons le temps à l'utilisateur pour regarder un nouveau secteur du
            Sleep(3000);
            //--- défilons le graphique de la position courante au bars_shift à droite
            ChartNavigate(0,CHART_CURRENT_POS,bars_shift);
           }
         //--- le changement du mode à l'opposé
         mode*=-1;
        }
      else // le mode du travail avec le bord droit du graphique
        {
         //--- défilons le graphique vers le bord droit
         ChartNavigate(0,CHART_END,pos);
         for(int i=0;i<pictures;i++)
           {
            //---préparons la signature sur le graphique et le nom pour le fichier
            string name="ChartScreenShot"+"CHART_END"+string(pos)+".gif";
            //--- la conclusion du nom sur le graphique en forme du commentaire
            Comment(name);
            //--- sauvegardons la capture d'écran au dossier le répertoire_du terminal
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_RIGHT))
               Print("Nous avons sauvegardé la capture d'écran ",name);
            //---
            pos+=bars_shift;
            //---donnons le temps à l'utilisateur pour regarder un nouveau secteur du
            Sleep(3000);
            //--- défilons le graphique de la position courante au bars_shift à droite
            ChartNavigate(0,CHART_CURRENT_POS,-bars_shift);
           }
         //--- le changement du mode à l'opposé
         mode*=-1;
        }
     }  // la fin du traitement de l'événement CHARTEVENT_CLICK
//--- la fin du gestionnaire OnChartEvent()
  }
```

Voir aussi

ChartNavigate(), Les ressources

© 2000-2015, MetaQuotes Software Corp.

Les fonctions commerciales

Le groupe des fonctions destinées à la gestion par l'activité commerciale.

Les fonctions commerciales peuvent être utilisées dans les experts et les scripts. Les fonctions commerciales peuvent être appelées seulement dans le cas où dans les propriétés de l'expert correspondant ou du script quand le point "Permettre au conseiller de commercer" est coché.

Разрешение или запрет на торговлю может зависеть от множества факторов, которые описаны в разделе "Разрешение на торговлю".

Fonction	Action
OrderCalcMargin	Calcule le montant de la marge nécessaire au type indiqué de l'ordre, en devise du compte
OrderCalcProfit	Calcule le montant du bénéfice en vertu des paramètres transmis en devise du compte
OrderCheck	Contrôle la suffisance des moyens pour l'accomplissement de la transaction commerciale demandée.
OrderSend	Expédie les demandes commerciales au serveur
OrderSendAsync	Envoie les demandes commerciales asynchronement sans attente de la réponse du serveur commercial
PositionsTotal	Rend le nombre de positions ouvertes
PositionGetSymbol	Rend le symbole de la position correspondante ouverte
PositionSelect	Choisit la position ouverte pour le travail ultérieur avec elle
PositionGetDouble	Rend la propriété demandée de la position ouverte (double)
PositionGetInteger	Rend la propriété demandée de la position ouverte (datetime ou int)
PositionGetString	Rend la propriété demandée de la position ouverte (string)
OrdersTotal	Rend le nombre d'ordres
OrderGetTicket	Rend le ticket de l'ordre correspondant
OrderSelect	Choisit l'ordre pour le travail ultérieur avec lui
OrderGetDouble	Rend la propriété demandée de l'ordre (double)
OrderGetInteger	Rend la propriété demandée de l'ordre (datetime ou int)

© 2000-2015, MetaQuotes Software Corp.

OrderGetString	Rend la propriété demandée de l'ordre (string)
HistorySelect	Demande l'histoire des marchés et des ordres pour la période indiquée du temps de serveur
HistorySelectByPosition	Demande l'histoire des marchés et des ordres avec l'identificateur indiqué de la position.
HistoryOrderSelect	Choisit à l'histoire l'ordre pour le travail ultérieur avec lui
HistoryOrdersTotal	Rend le nombre d'ordres dans l'histoire
HistoryOrderGetTicket	Rend le ticket de l'ordre correspondant dans l'histoire
HistoryOrderGetDouble	Rend la propriété demandée de l'ordre dans l'histoire (double)
HistoryOrderGetInteger	Rend la propriété demandée de l'ordre dans l'histoire (datetime ou int)
HistoryOrderGetString	Rend la propriété demandée de l'ordre dans l'histoire (string)
HistoryDealSelect	Choisit le marché dans l'histoire pour davantage l'appeler par les fonctions appropriées
HistoryDealsTotal	Rend le nombre de marchés dans l'histoire
HistoryDealGetTicket	Choisit le marché pour le traitement ultérieur et rend le ticket les marchés dans l'histoire
HistoryDealGetDouble	Rend la propriété demandée du marché dans l'histoire (double)
HistoryDealGetInteger	Rend la propriété demandée du marché dans l'histoire (datetime ou int)
HistoryDealGetString	Rend la propriété demandée du marché dans l'histoire (string)

© 2000-2015, MetaQuotes Software Corp.

OrderCalcMargin

Calcule le montant de la marge nécessaire au type indiqué de l'ordre sur le compte courant et à l'environnement courant de marché sans prendre en compte des ordres remis et des positions ouvertes. Permet d'évaluer le montant de la marge pour la transaction planifiée commerciale. La valeur revient en devise du compte.

```
bool  OrderCalcMargin(
   ENUM_ORDER_TYPE       action,           // le type de l'ordre
   string                symbol,           // le nom du symbole
   double                volume,           // le volume
   double                price,            // le prix de l'ouverture
   double&               margin            // la variable pour la réception de la vale
   );
```

Les paramètres

action

[in] Le type de l'ordre peut prendre les valeurs de l'énumération ENUM_ORDER_TYPE.

symbol

[in] Le nom de l'instrument financier.

volume

[in] Le volume de la transaction commerciale.

price

[En] Le prix de l'ouverture.

margin

[out] La variable, où on inscrit le montant nécessaire de la marge en cas de l'exécution réussie de la fonction. Le calcul est produit comme si sur le compte courant il n'y avait pas des ordres remis et des positions ouvertes. La valeur de la marge dépend de plusieurs facteurs et peut changer au changement de l'environnement de marché.

La valeur rendue

Rend true en cas du succès, autrement rend false. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Voir aussi

OrderSend(), Les propriétés des ordres, Les types des transactions commerciales

© 2000-2015, MetaQuotes Software Corp.

OrderCalcProfit

Calcule le montant du bénéfice pour le compte courant et l'environnement de marché sur la base des paramètres transmis. Est destinée à l'appréciation préalable du résultat de la transaction commerciale. La valeur revient en devise du compte.

```
bool  OrderCalcProfit(
   ENUM_ORDER_TYPE       action,          // le type de l'ordre (ORDER_TYPE_BUY ou OR
   string                symbol,          // le nom du symbole
   double                volume,          // le volume
   double                price_open,      // le prix de l'ouverture
   double                price_close,     // le prix de la clôture
   double&               profit           // la variable pour la réception de la valeu
   );
```

Les paramètres

action

[in] Le type de l'ordre peut accepter une de deux valeurs de l'énumération ENUM_ORDER_TYPE: ORDER_TYPE_BUY ou ORDER_TYPE_SELL.

symbol

[in] Le nom de l'instrument financier.

volume

[in] Le volume de la transaction commerciale.

price_open

[En] Le prix de l'ouverture.

price_close

[in] Le prix de la clôture.

profit

[out] La variable, où on inscrit la valeur calculée du bénéfice en cas de l'exécution réussie de la fonction. La valeur de l'évaluation du bénéfice dépend de plusieurs facteurs et peut changer au changement de l'environnement de marché.

La valeur rendue

Rend true en cas du succès, autrement rend false. Si on indique le type inadmissible de l'ordre, la fonction rendra false. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Voir aussi

OrderSend(), Les propriétés des ordres, Les types des transactions commerciales

© 2000-2015, MetaQuotes Software Corp.

OrderCheck

La fonction OrderCheck() vérifie la suffisance des moyens pour l'accomplissement de la transaction commerciale exigée. Les résultats du contrôle sont placés aux champs de la structureMqlTradeCheckResult.

```
bool  OrderCheck(
   MqlTradeRequest&       request,      // la structure de la demande
   MqlTradeCheckResult&   result        // la structure de la réponse
   );
```

Les paramètres

request

[in] L'indicateur sur la structure du type MqlTradeRequest, qui décrit l'action commerciale demandée.

result

[in,out] L'indicateur sur la structure du type MqlTradeCheckResult, où sera placé le résultat du contrôle.

La valeur rendue

En cas du manque des moyens ou des paramètres erronément remplis la fonction rend false. En cas du contrôle de base réussi des structures (le contrôle des indicateurs) revient true - **ce n'est pas une indication que la transaction commerciale demandée sera accomplie avec succès**. Pour la réception de la description détaillée du résultat de l'exécution de la fonction il faut analyser les champs de la structure *result*.

Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Voir aussi

OrderSend(), Les types des transactions commerciales, La structure de la demande commerciale, La structure des résultats du contrôle de la demande commerciale, La structure du résultat de la demande commerciale

© 2000-2015, MetaQuotes Software Corp.

OrderSend

La fonction OrderSend() est destinée à la réalisation de l'activité commerciale dans le cadre de MQL5.

```
bool  OrderSend(
   MqlTradeRequest&  request      // structure de la demande
   MqlTradeResult&   result       // structure de la réponse
   );
```

Paramètres

request

[in] L'indicateur sur la structure du type MqlTradeRequest, décrivant l'activité commerciale du client.

result

[in,out] L'indicateur sur la structure du type MqlTradeResult, décrivant le résultat de la transaction commerciale en cas de l'exécution réussie (du retour true).

La valeur rendue

En cas du contrôle réussi de base des structures (le contrôle des indicateurs)true revient - **cela ne témoigne pas de l'exécution réussie de la transaction commerciale**. Pour la réception de la description plus détaillée du résultat de l'exécution de la fonction il faut analyser les champs de la structure *result*.

Note

La demande commerciale passe quelques stades de contrôle sur le serveur commercial. On contrôle en premier lieu l'exactitude du remplissage de tous les champs nécessaires du paramètre *request* , et en l'absence des erreurs le serveur accepte l'ordre pour le traitement ultérieur. A l'acceptation réussie de l'ordre par le serveur commercial la fonction OrderSend() rend la valeur true.

Il est recommandé de contrôler de soi-même la demande devant l'expédition à son serveur commercial. Pour le contrôle de la demande il y a la fonction OrderCheck(), qui non seulement contrôlera la suffisance des moyens pour l'accomplissement de la transaction commerciale, mais aussi rendra beaucoup d'autres options utilesaux résultats de la vérification de la demande commerciale:

- le code du retour, qui signalera l'erreur dans la demande testée;
- la valeur de la balance, qui sera après l'exécution de la transaction commerciale;
- la valeur des moyens propres, qui sera après l'exécution de la transaction commerciale;
- la valeur du bénéfice flottant, qui sera après l'exécution de la transaction commerciale;
- le montant de la marge nécessaire à la transaction demandée commerciale;
- le montant des moyens propres libres, qui resteront après l'exécution de la transaction demandée commerciale;
- le niveau de la marge, qui s'établira après l'exécution de la transaction demandée commerciale;
- le commentaire sur le code de la réponse, la description de l'erreur.

Il faut avoir en vue à l'exposition de l'ordre de marché que la fin réussie du travail de la méthode OrderSend() ne signifie pas toujours l'accomplissement réussi du marché. Il est nécessaire de contrôler dans la structure rendue du résultat *result* la valeur *retcode*, contenante le code du retour du serveur commercial, ainsi que la valeur des champs *deal* ou *order* en fonction du type de

© 2000-2015, MetaQuotes Software Corp.

l'opération.

Chaque ordre accepté se trouve sur le serveur commercial en attendant le traitement jusqu'à ce qu'arrive une des conditions pour son exécution:

- l'expiration du délai de l'action,
- l'apparition de la demande de répétition,
- le fonctionnement de l'ordre à l'entrée du prix de l'exécution,
- l'entrée de la demande sur l'annulation de l'ordre.

Au moment du traitement de l'ordre le serveur commercial envoie au terminal le message sur l'arrivée de l'événement commercial Trade, qu'on peut traiter par la fonction OnTrade().

Le résultat de l'exécution de la demande commerciale sur le serveur, envoyée par la fonction OrderSend() peut être contrôlée à l'aide d'un gestionnaire OnTradeTransaction. En cela il faut prendre en considération qu'à la suite de l'exécution d'une demande commerciale le gestionnaire OnTradeTransaction sera appelé plusieurs fois.

Par exemple, à l'envoi de l'ordre de marché sur l'achat, il est traité, l'ordre correspondant sur l'achat est créé pour le compte, se passe l'exécution de l'ordre, son suppression de la liste des ouverts, l'ajout à l'histoire des ordres, ensuite le marché correspondant est ajouté à l'histoire et une nouvelle position est créé. Pour chacun de ces événements sera appelée la fonction OnTradeTransaction.

Exemple:

```
//--- valeurs pour ORDER_MAGIC
input long order_magic=55555;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- nous persuaderons que le compte est celui d'apprentissage
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("Travail du script au compte réel est interdit!");
      return;
     }
//--- établissons ou supprimons l'ordre
   if(GetOrdersTotalByMagic(order_magic)==0)
     {
      //--- il n'y a pas d'ordres courants - établissons l'ordre
      uint res=SendRandomPendingOrder(order_magic);
      Print("Code de retour du serveur commercial ",res);
     }
   else // il y a des ordres - supprimons les ordres
     {
      DeleteAllOrdersByMagic(order_magic);
     }
//---
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Recevoir le nombre courant d'ordres avec indiqué ORDER_MAGIC     |
//+------------------------------------------------------------------+
int GetOrdersTotalByMagic(long const magic_number)
  {
   ulong order_ticket;
   int total=0;
//--- passerons selon les ordres remis
   for(int i=0;i<OrdersTotal();i++)
      if((order_ticket=OrderGetTicket(i))>0)
         if(magic_number==OrderGetInteger(ORDER_MAGIC)) total++;
//---
   return(total);
  }
//+------------------------------------------------------------------+
//| Supprime tous les ordres remis avec indiqué ORDER_MAGIC          |
//+------------------------------------------------------------------+
void DeleteAllOrdersByMagic(long const magic_number)
  {
   ulong order_ticket;
//--- passerons selon les ordres remis
   for(int i=0;i<OrdersTotal();i++)
      if((order_ticket=OrderGetTicket(i))>0)
         //--- ordre avec le convenant ORDER_MAGIC
         if(magic_number==OrderGetInteger(ORDER_MAGIC))
           {
            MqlTradeResult result={0};
            MqlTradeRequest request={0};
            request.order=order_ticket;
            request.action=TRADE_ACTION_REMOVE;
            OrderSend(request,result);
            //--- sortons dans le log la réponse du serveur
            Print(__FUNCTION__,": ",result.comment," code de la réponse ",result.retco
           }
//---
  }
//+------------------------------------------------------------------+
//| Placer l'ordre remis par hasard                                  |
//+------------------------------------------------------------------+
uint SendRandomPendingOrder(long const magic_number)
  {
//--- préparons la requête
   MqlTradeRequest request={0};
   request.action=TRADE_ACTION_PENDING;         // installation de l'ordre remis
   request.magic=magic_number;                  // ORDER_MAGIC
   request.symbol=_Symbol;                      // instrument
   request.volume=0.1;                          // volume à 0.1 lot
   request.sl=0;                                // Stop Loss n'a pas indiqué
   request.tp=0;                                // Take Profit n'a pas indiqué
//--- formerons le type de l'ordre
```

© 2000-2015, MetaQuotes Software Corp.

```
   request.type=GetRandomType();                  // type de l'ordre
//---formerons le prix pour l'ordre remis
   request.price=GetRandomPrice(request.type);  // prix pour l'ouverture
//--- expédierons l'ordre commercial
   MqlTradeResult result={0};
   OrderSend(request,result);
//---sortons dans le log la réponse du serveur
   Print(__FUNCTION__,":",result.comment);
   if(result.retcode==10016) Print(result.bid,result.ask,result.price);
//--- rendons le code de la réponse du serveur commercial
   return result.retcode;
  }
//+------------------------------------------------------------------+
//| Recevoir le type du type remis occasionnellement                 |
//+------------------------------------------------------------------+
ENUM_ORDER_TYPE GetRandomType()
  {
   int t=MathRand()%4;
//---   0<=t<4
   switch(t)
     {
      case(0):return(ORDER_TYPE_BUY_LIMIT);
      case(1):return(ORDER_TYPE_SELL_LIMIT);
      case(2):return(ORDER_TYPE_BUY_STOP);
      case(3):return(ORDER_TYPE_SELL_STOP);
     }
//--- valeur inadmissible
   return(WRONG_VALUE);
  }
//+------------------------------------------------------------------+
//| Recevoir le prix occasionnellement                               |
//+------------------------------------------------------------------+
double GetRandomPrice(ENUM_ORDER_TYPE type)
  {
   int t=(int)type;
//--- niveau des arrêts selon le caractère
   int distance=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
//--- recevrons les données du dernier tick
   MqlTick last_tick={0};
   SymbolInfoTick(_Symbol,last_tick);
//---  évaluerons le prix conformément au type
   double price;
   if(t==2 || t==5) // ORDER_TYPE_BUY_LIMIT ou ORDER_TYPE_SELL_STOP
     {
      price=last_tick.bid; // nous répoussons du prix Bid
      price=price-(distance+(MathRand()%10)*5)*_Point;
     }
   else             // ORDER_TYPE_SELL_LIMIT ou ORDER_TYPE_BUY_STOP
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      price=last_tick.ask; // nous répoussons du prix Ask
      price=price+(distance+(MathRand()%10)*5)*_Point;
     }
//---
   return(price);
  }
```

Voir aussi

Types des transactions commerciales, Structure de la requête commerciale, Structure du résultat de la requête commerciale

© 2000-2015, MetaQuotes Software Corp.

OrderSendAsync

La fonction OrderSendAsync() est destinée aux opérations commerciales asynchrones sans attente de la réponse du serveur commercial à la demande envoyée. La fonction est destinée au commerce à haute fréquence, quand il est inadmissible perdre le temps à l'attente de la réponse du serveur selon les conditions de l'algorithme commercial .

```
bool  OrderSendAsync(
   MqlTradeRequest&  request,      // la structure de la demande
   MqlTradeResult&   result        // la structure de la réponse
   );
```

Paramètres

request

[in] L'indicateur sur la structure du type MqlTradeRequest,décrivant l'action commerciale du client.

result

[in,out] L'indicateur sur la structure du type MqlTradeResult, décrivant le résultat de l'opération commerciale en cas de l'exécution réussie de la fonction (au retour true).

La valeur rendue

Rend true du fait de l'expédition de la demande au serveur commercial. Si la demande n'a pas été envoyé, on rend false. A l'exécution fructueuse dans la variable *result*le code de la réponse contient la signification TRADE_RETCODE_PLACED (le code 10008) - "l'ordre est installé". L'exécution réussie signifie seulement le fait de l'envoi, mais ne donne pas aucune garantie que la demande est arrivée au serveur commercial et a été acceptée pour le traitement. Le serveur commercial au traitement de la demande reçue expédie au terminal de client le message de réponse sur le changement de l'état courant des positions, des ordres et des marchés, qui amène à la génération de l'événement Trade.

Le résultat de l'exécution de la demande commerciale sur le serveur, envoyée par la fonction OrderSendAsync() peut être contrôlée à l'aide d'un gestionnaire OnTradeTransaction. En cela il faut prendre en considération qu'à la suite de l'exécution d'une demande commerciale le gestionnaire OnTradeTransaction sera appelé plusieurs fois.

Par exemple, à l'envoi de l'ordre de marché sur l'achat, il est traité, l'ordre correspondant sur l'achat est créé pour le compte, se passe l'exécution de l'ordre, son suppression de la liste des ouverts, l'ajout à l'histoire des ordres, ensuite le marché correspondant est ajouté à l'histoire et une nouvelle position est créé. Pour chacun de ces événements sera appelée la fonction OnTradeTransaction. Pour la réception de l'information détaillée il faut analyser les paramètres de cette fonction:

- **trans** -dans ce paramètre est transmise la structure MqlTradeTransaction, décrivant la transaction commerciale, appliquée au compte commercial;
- **request** - dans ce paramètre est transmise la structureMqlTradeRequest, décrivant la demande commerciale, à la suite de laquelle est exécutée la transaction commerciale;
- **result** -dans ce paramètre est transmise la structure MqlTradeResult, décrivant le résultat de l'exécution de la demande commerciale.

Note

La fonction selon son assignement et ses paramètres est analogue àOrderSend(),mais elle est la

© 2000-2015, MetaQuotes Software Corp.

version asynchrone, c'est-à-dire n'arrête pas le travail du programme dans l'attente du résultat de son exécution. On peut comparer la vitesse des opérations commerciales de ces deux fonctions à l'aide du conseiller donné dans l'exemple.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
#property description "L'expert pour l'expédition des demandes commerciales "
                      "avec l'utilisation de la fonction OrderSendAsync().\r\n"
#property description "On montre le traitement des événements commerciaux à l'aide"
                      " des fonctions - gestionnaires OnTrade() et OnTradeTransaction
#property description "Dans les paramètres de l'expert on peut spécifier Magic Number'
                      " (l'identificateur unique) "
#property description "et le régime de la sortie des messages au journal "Experts". Or
//--- input parameters
input int  MagicNumber=1234567;      // L'identificateur de l'expert
input bool DescriptionModeFull=true; // Le régime de la sortie détaillée
//--- la variable pour l'utilisation dans l'appel HistorySelect()
datetime history_start;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vérifions l'autorisation à la négociation automatique
   if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
     {
      Alert("La négociation automatique est interdite dans le terminal, l'expert sera
      ExpertRemove();
      return(-1);
     }
//--- il est interdit de faire le commerce sur le compte réel!
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("il est interdit au conseiller de vendre sur le compte réel!");
      ExpertRemove();
      return(-2);
     }
//--- si on peut faire le commerce sur ce compte (par exemple c'est impossible sous le
   if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
     {
      Alert("Le commerce à ce compte est interdit");
      ExpertRemove();
      return(-3);
     }
//--- rappelons le temps du lancement de l'expert pour la réception de l'histoire comm
   history_start=TimeCurrent();
//---
   CreateBuySellButtons();
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- supprimons les objets graphiques après soi-même
   ObjectDelete(0,"Buy");
   ObjectDelete(0,"Sell");
//---
  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//---le titre selon le nom de la fonction-gestionnaire de l'événement commercial
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- recevrons le type de la transaction en forme de la valeur de l'énumération
   ENUM_TRADE_TRANSACTION_TYPE type=trans.type;
//--- si la transaction est le résultat du traitement de la demande
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      //---déduisons le nom de la transaction
      Print(EnumToString(type));
      //--- Puis nous déduisons la description de chaîne de la demande traitée
      Print("------------RequestDescription\r\n",
            RequestDescription(request,DescriptionModeFull));
      //--- et déduisons la description du résultat de la demande
      Print("------------ ResultDescription\r\n",
            TradeResultDescription(result,DescriptionModeFull));
     }
   else //déduisons la description complète de la transaction pour les transactions d'
     {
      Print("------------ TransactionDescription\r\n",
            TransactionDescription(trans,DescriptionModeFull));
     }
//---
  }
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- Les membres statiques pour stocker l'état du compte
   static int prev_positions=0,prev_orders=0,prev_deals=0,prev_history_orders=0;
//--- demanderons l'histoire commerciale
   bool update=HistorySelect(history_start,TimeCurrent());
   PrintFormat("HistorySelect(%s , %s) = %s",
               TimeToString(history_start),TimeToString(TimeCurrent()),(string)update);
//--- le titre selon le nom de la fonction-gestionnaire de l'événement commercial
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- déduisons le nom du gestionnaire et le nombre d'ordres au moment du traitement
   int curr_positions=PositionsTotal();
   int curr_orders=OrdersTotal();
   int curr_deals=HistoryOrdersTotal();
   int curr_history_orders=HistoryDealsTotal();
//---déduisons le nombre d'ordres, des positions, des marchés, ainsi que le changement
   PrintFormat("PositionsTotal() = %d (%+d)",
               curr_positions,(curr_positions-prev_positions));
   PrintFormat("OrdersTotal() = %d (%+d)",
               curr_orders,curr_orders-prev_orders);
   PrintFormat("HistoryOrdersTotal() = %d (%+d)",
               curr_deals,curr_deals-prev_deals);
   PrintFormat("HistoryDealsTotal() = %d (%+d)",
               curr_history_orders,curr_history_orders-prev_history_orders);
//---la rupture de la ligne pour faciliter la lecture du Journal
   Print("");
//--- rappelons l'état du compte
   prev_positions=curr_positions;
   prev_orders=curr_orders;
   prev_deals=curr_deals;
   prev_history_orders=curr_history_orders;
//---
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- le traitement de l'événement CHARTEVENT_CLICK ("La pression du bouton de la sou
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("=> ",__FUNCTION__,": sparam = ",sparam);
      //--- le volume minimal pour le marché
      double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
      //--- si le bouton "Buy" est appuyé, alors on achète
      if(sparam=="Buy")
        {
         PrintFormat("Buy %s %G lot",_Symbol,volume_min);
         BuyAsync(volume_min);
         //--- appuyez sur le bouton pressé encore une fois
         ObjectSetInteger(0,"Buy",OBJPROP_STATE,false);
        }
      //--- Si le bouton "Sell" est appuyé, vendons
      if(sparam=="Sell")
        {
         PrintFormat("Sell %s %G lot",_Symbol,volume_min);
         SellAsync(volume_min);
         //--- appuyez sur le bouton pressé encore une fois
         ObjectSetInteger(0,"Sell",OBJPROP_STATE,false);
        }
      ChartRedraw();
     }
//---
  }
//+------------------------------------------------------------------+
//| Rend la description de texte de la transaction                   |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans,
                              const bool detailed=true)
  {
//--- préparerons la chaîne pour le retour de la fonction
   string desc=EnumToString(trans.type)+"\r\n";
//--- en régime détaillé ajoutons le maximum de l'information
   if(detailed)
     {
      desc+="Symbol: "+trans.symbol+"\r\n";
      desc+="Deal ticket: "+(string)trans.deal+"\r\n";
      desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
      desc+="Order ticket: "+(string)trans.order+"\r\n";
      desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
      desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
      desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
      desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
      desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
      desc+="Price trigger: "+StringFormat("%G",trans.price_trigger)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
      desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
     }
//--- rendons la chaîne reçue
   return desc;
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Rend la description de texte de la demande commerciale           |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request,
                          const bool detailed=true)
  {
//--- préparerons la chaîne pour le retour de la fonction
   string desc=EnumToString(request.action)+"\r\n";
//--- en régime détaillé ajoutons le maximum de l'information
   if(detailed)
     {
      desc+="Symbol: "+request.symbol+"\r\n";
      desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
      desc+="Order ticket: "+(string)request.order+"\r\n";
      desc+="Order type: "+EnumToString(request.type)+"\r\n";
      desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
      desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
      desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
      desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
      desc+="Deviation points: "+StringFormat("%G",request.deviation)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
      desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
      desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
      desc+="Comment: "+request.comment+"\r\n";
     }
//--- rendons la chaîne reçue
   return desc;
  }
//+------------------------------------------------------------------+
//| Rend la description de texte du résultat du traitement de la demande |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result,
                              const bool detailed=true)
  {
//--- préparerons la chaîne pour le retour de la fonction
   string desc="Retcode "+(string)result.retcode+"\r\n";
//--- en régime détaillé ajoutons le maximum de l'information
   if(detailed)
     {
      desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
      desc+="Order ticket: "+(string)result.order+"\r\n";
      desc+="Deal ticket: "+(string)result.deal+"\r\n";
      desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
      desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
      desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
      desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
      desc+="Comment: "+result.comment+"\r\n";
     }
//--- rendons la chaîne reçue
   return desc;
  }
//+------------------------------------------------------------------+
//| Crée deux boutons pour l'achat et la vente                       |
//+------------------------------------------------------------------+
void CreateBuySellButtons()
  {
//---vérifions la présence de l'objet avec le nom "Buy"
   if(ObjectFind(0,"Buy")>=0)
     {
      //---Si l'objet trouvé n'est pas le bouton - supprimons-le
      if(ObjectGetInteger(0,"Buy",OBJPROP_TYPE)!=OBJ_BUTTON)
```

© 2000-2015, MetaQuotes Software Corp.

```
         ObjectDelete(0,"Buy");
     }
   else
      ObjectCreate(0,"Buy",OBJ_BUTTON,0,0,0); // créons le bouton "Buy"
//--- configurons le button "Buy"
   ObjectSetInteger(0,"Buy",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Buy",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Buy",OBJPROP_YDISTANCE,50);
   ObjectSetInteger(0,"Buy",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Buy",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Buy",OBJPROP_TEXT,"Buy");
   ObjectSetInteger(0,"Buy",OBJPROP_COLOR,clrRed);
//---vérifions la présence de l'objet avec le nom "Sell"
   if(ObjectFind(0,"Sell")>=0)
     {
      //--- si l'objet trouvé n'est pas le bouton - supprimons-le
      if(ObjectGetInteger(0,"Sell",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Sell");
     }
   else
      ObjectCreate(0,"Sell",OBJ_BUTTON,0,0,0); // créons le bouton "Sell"
//--- configurons le button "Sell"
   ObjectSetInteger(0,"Sell",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Sell",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Sell",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Sell",OBJPROP_TEXT,"Sell");
   ObjectSetInteger(0,"Sell",OBJPROP_COLOR,clrBlue);
//--- forcément mettons à jour le graphique pour que les boutons sont déssinés immédia
   ChartRedraw();
//---
  }
//+------------------------------------------------------------------+
//| L'achat par la fonction asynchrone OrderSendAsync()              |
//+------------------------------------------------------------------+
void BuyAsync(double volume)
  {
//--- préparerons la demande
   MqlTradeRequest req={0};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_BUY;
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_ASK);
   req.deviation   =10;
   req.comment     ="Buy using OrderSendAsync()";
   MqlTradeResult  res={0};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": l'erreur ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
//| La vente par la fonction asynchrone OrderSendAsync()             |
//+------------------------------------------------------------------+
void SellAsync(double volume)
  {
//--- préparerons la demande
```

© 2000-2015, MetaQuotes Software Corp.

```
   MqlTradeRequest req={0};
   req.action       =TRADE_ACTION_DEAL;
   req.symbol       =_Symbol;
   req.magic        =MagicNumber;
   req.volume       =0.1;
   req.type         =ORDER_TYPE_SELL;
   req.price        =SymbolInfoDouble(req.symbol,SYMBOL_BID);
   req.deviation    =10;
   req.comment      ="Sell using OrderSendAsync()";
   MqlTradeResult  res={0};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": l'erreur ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
```

La sortie approximative des messages au journal "Experts":

© 2000-2015, MetaQuotes Software Corp.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnChartEvent: sparam = Sell
12:52:52   ExpertAdvisor (EURUSD,H1)   Sell EURUSD 0.01 lot
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_REQUEST
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------RequestDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_ACTION_DEAL
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Magic Number: 1234567
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order filling: ORDER_FILLING_FOK
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Deviation points: 10
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Limit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Comment: Sell using OrderSendAsync()
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ ResultDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   Retcode 10009
12:52:52   ExpertAdvisor (EURUSD,H1)   Request ID: 2
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Ask: 1.29319
12:52:52   ExpertAdvisor (EURUSD,H1)   Bid: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Comment:
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+1)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_DELETE
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
```

© 2000-2015, MetaQuotes Software Corp.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_HISTORY_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_FILLED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_DEAL_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
```

© 2000-2015, MetaQuotes Software Corp.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
```

© 2000-2015, MetaQuotes Software Corp.

PositionsTotal

Rend le nombre de positions ouvertes.

```
int  PositionsTotal();
```

La valeur rendue

La valeur du type int.

Note

Pour chaque symbole à tout moment on ne peut ouvrir qu'une seule position, qui est le résultat d'un ou plusieurs marchés. Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments".

Le total des positions sur le compte commercial ne peut excéder le nombre total d'instruments financiers.

Voir aussi

PositionGetSymbol(), PositionSelect(), Propriété des positions

© 2000-2015, MetaQuotes Software Corp.

PositionGetSymbol

Rend le symbole de la position correspondante ouverte et Choisit automatiquement la position pour le travail ultérieur avec elle à l'aide des fonctions PositionGetDouble, PositionGetInteger, PositionGetString.

```
string  PositionGetSymbol(
   int  index      // numéro dans la liste des positions
   );
```

Paramètres

index

[in] Le numéro de la position dans la liste des positions ouvertes.

La valeur rendue

La valeur du type string. Si la position n'est pas trouvée, une chaîne vide reviendra. Pour recevoir le code de l'erreur il faut appeler la fonction GetLastError().

Note

Pour chaque symbole à tout moment on ne peut ouvrir qu'une seule position, qui est le résultat d'un ou plusieurs marchés. Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes,, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments".

Le total des positions sur le compte commercial ne peut excéder le nombre total d'instruments financiers.

Voir aussi

PositionsTotal(), PositionSelect(), Propriété des positions

© 2000-2015, MetaQuotes Software Corp.

PositionSelect

Choisit la position ouverte pour le travail ultérieur avec elle. Rend true à l'achèvement réussi de la fonction. Rend false à l'achèvement raté de la fonction. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

```
bool  PositionSelect(
   string  symbol,     // nom de l'instrument
   );
```

Paramètres

symbol

[in] Le nom de l'instrument financier.

La valeur rendue

La valeur du type bool.

Note

Pour chaque symbole à tout moment on ne peut ouvrir qu'une seule position, qui est le résultat d'un ou plusieursmarchés. Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes,, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments" du terminal de client.

La fonction PositionSelect() copie les données sur l'ordre à l'environnement logiciel et les appels ultérieurs PositionGetDouble(), PositionGetInteger() et PositionGetString() rendent les données auparavant copiées. Cela signifie que la position même ne peut plus exister (ou elle a changé par le volume, la direction etc.), et on peut encore recevoir les données de cette position. Pour la réception garantie des données nouvelles sur la position il est recommandé d'appeler la fonction PositionSelect() avant de les appeler directement.

Voir aussi

PositionGetSymbol(), PositionsTotal(), Propriété des positions

© 2000-2015, MetaQuotes Software Corp.

PositionGetDouble

La fonction rend la propriété demandée de la position ouverte préalablement choisie à l'aide de la fonction PositionGetSymbol ou PositionSelect. La propriété de la position doit être du type double. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
double  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id      // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id,     // identificateur de la propriété
   double&                        double_var       // acceptons ici la valeur de la pr
   );
```

Paramètres

property_id

[in] Rend directement la valeur de la propriété. La valeur peut être une des valeurs de l'énumération ENUM_POSITION_PROPERTY_DOUBLE.

double_var

[out] La variable du type double, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type double.

Note

Pour chaque symbole à tout moment on ne peut ouvrir qu'une seule position, qui est le résultat d'un ou plusieursmarchés. Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes,, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments" du terminal de client.

Pour la réception garantie des données nouvelles sur la position il est recommandé d'appeler la fonction PositionSelect() avant de les appeler directement.

Voir aussi

PositionGetSymbol(), PositionSelect(), Propriété des positions

© 2000-2015, MetaQuotes Software Corp.

PositionGetInteger

La fonction rend la propriété demandée de la position ouverte préalablement choisie à l'aide de la fonction PositionGetSymbol ou PositionSelect. La propriété de la position doit être du type datetime, int.Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
long  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id      // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id,     // identificateur de la propriété
   long&                           long_var         // acceptons ici la valeur de la p
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété de la position. La valeur peut être une des valeurs de l'énumération ENUM_POSITION_PROPERTY_INTEGER.

long_var

[out] La variable du type long, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type long.

Note

Pour chaque symbole à tout moment on ne peut ouvrir qu'une seule position, qui est le résultat d'un ou plusieursmarchés. Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes,, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments" du terminal de client.

Pour la réception garantie des données nouvelles sur la position il est recommandé d'appeler la fonction PositionSelect() avant de les appeler directement.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- vérifions la présence de la position et déduisons le temps de son changement
   if(PositionSelect(_Symbol))
     {
//--- recevrons l'identificateur de la position pour un travail ultérieur avec elle
      ulong position_ID=PositionGetInteger(POSITION_IDENTIFIER);
      Print(_Symbol," postion #",position_ID);
//--- recevrons le temps de la formation de la position dans les millisecondes de 01.0
      long create_time_msc=PositionGetInteger(POSITION_TIME_MSC);
      PrintFormat("Position #%d  POSITION_TIME_MSC = %i64 milliseconds => %s",positior
                  create_time_msc,TimeToString(create_time_msc/1000));
//--- recevrons le temps du dernier changement de la position en secondes de 01.01.197
      long update_time_sec=PositionGetInteger(POSITION_TIME_UPDATE);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE = %i64 seconds => %s",
                  position_ID,update_time_sec,TimeToString(update_time_sec));
//--- recevrons le temps du dernier changement de la position en millisecondes de 01.0
      long update_time_msc=PositionGetInteger(POSITION_TIME_UPDATE_MSC);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE_MSC = %i64 milliseconds => %s",
                  position_ID,update_time_msc,TimeToString(update_time_msc/1000));
     }
//---
  }
```

Voir aussi

PositionGetSymbol(), PositionSelect(), Propriété des positions

© 2000-2015, MetaQuotes Software Corp.

PositionGetString

La fonction rend la propriété demandée de la position ouverte préalablement choisie à l'aide de la fonction PositionGetSymbol ou PositionSelect. La propriété de la position doit être du type string. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
string  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id      // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id,     // identificateur de la propriété
   string&                        string_var       // acceptons ici la valeur de la pr
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété de la position. La valeur peut être une des valeurs de l'énumération ENUM_POSITION_PROPERTY_STRING.

string_var

[out] La variable du type string, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type string.

Note

Pour chaque symbole à tout moment on ne peut ouvrir qu'une seule position, qui est le résultat d'un ou plusieursmarchés. Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes,, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments" du terminal de client.

Pour la réception garantie des données nouvelles sur la position il est recommandé d'appeler la fonction PositionSelect() avant de les appeler directement.

Voir aussi

PositionGetSymbol(), PositionSelect(), Propriété des positions

© 2000-2015, MetaQuotes Software Corp.

OrdersTotal

Rend le nombre d'ordres.

```
int  OrdersTotal();
```

La valeur rendue

La valeur du type int.

Note

Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments". L'ordre- c'est une disposition sur la réalisation de la transaction commerciale, et la position c'est le résultat d'un ou plusieurs marchés.

Pour chaque symbole à tout moment on ne peut ouvrir qu'une seule position, tandis que on peut avoir plusieurs ordres remis selon le même symbole.

Voir aussi

OrderSelect(), OrderGetTicket(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

OrderGetTicket

Rend le ticket de l'ordre correspondant et choisit automatiquement l'ordre pour le travail ultérieur avec lui à l'aide des fonctions.

```
ulong  OrderGetTicket(
   int  index      // numéro dans la liste des ordres
   );
```

Paramètres

index

[in] Le numéro d'ordre dans la liste des ordres.

La valeur rendue

La valeur du type ulong. En cas d'une mauvaise exécution rend 0.

Note

Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments". L'ordre- c'est une disposition sur la réalisation de la transaction commerciale, et la position c'est le résultat d'un ou plusieurs marchés.

Pour chaque symbole à tout moment on ne peut ouvrir qu'une seule position, tandis que on peut avoir plusieurs ordres remis selon le même symbole.

La fonction OrderGetTicket() copie les données sur l'ordre à l'environnement logiciel et les appels ultérieurs OrderGetDouble(), OrderGetInteger(), OrderGetString() rendent les données auparavant copiées. Cela signifie que l'ordre même ne peut plus exister (ou le prix d'ouverture, les niveaux Stop Loss / Take Profit ou le moment de l'expiration ont y changé), et cet ordre peut encore recevoir les données. Pour la réception garantie des données nouvelles sur l'ordre il est recommandé d'appeler la fonction OrderGetTicket () avant de les appeler directement.

Exemple:

```
void OnStart()
  {
//--- les variables pour la réception des valeurs des propriétés de l'ordre
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- nombre d'ordres courants remis
   uint     total=OrdersTotal();
//--- passerons dans la boucle selon tous les ordres
   for(uint i=0;i<total;i++)
     {
      //--- recevrons le ticket les ordres à sa position dans le répertoire
```

© 2000-2015, MetaQuotes Software Corp.

```
      if((ticket=OrderGetTicket(i))>0)
        {
         //--- recevrons les propriétés de l'ordre
         open_price    =OrderGetDouble(ORDER_PRICE_OPEN);
         time_setup    =(datetime)OrderGetInteger(ORDER_TIME_SETUP);
         symbol        =OrderGetString(ORDER_SYMBOL);
         order_magic   =OrderGetInteger(ORDER_MAGIC);
         positionID    =OrderGetInteger(ORDER_POSITION_ID);
         initial_volume=OrderGetDouble(ORDER_VOLUME_INITIAL);
         type          =EnumToString(ENUM_ORDER_TYPE(OrderGetInteger(ORDER_TYPE)));
         //--- préparerons et sortons l'information sur l'ordre
         printf("#ticket %d %s %G %s at %G was set up at %s",
                ticket,                  // ticket d'ordre
                type,                    // type
                initial_volume,          // volume exposé
                symbol,                  // symbole, selon qui ont exposé
                open_price,              // prix indiqué de l'ouverture
                TimeToString(time_setup)// temps de l'installation de l'ordre
                );
        }
     }
//---
  }
```

Voir aussi

OrdersTotal(), OrderSelect(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

OrderSelect

Choisit l'ordre pour le travail ultérieur avec lui. Rend true à l'achèvement réussi de la fonction. Rend false à l'achèvement raté de la fonction. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

```
bool  OrderSelect(
   ulong    ticket,      // ticket d'ordre
   );
```

Paramètres

ticket

[in] Le ticket d'ordre.

La valeur rendue

La valeur du type bool.

Note

Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments" du terminal de client. Pour chaque symbole à tout moment on ne peut ouvrir qu'une position , tandis qu'on peut avoir plusieurs ordres remis selon le même symbole.

La fonction OrderSelect() copie les données sur l'ordre à l'environnement logiciel et les appels ultérieurs OrderGetDouble(), OrderGetInteger(), OrderGetString() rendent les données auparavant copiées. Cela signifie que l'ordre même ne peut plus exister (ou le prix d'ouverture, les niveaux Stop Loss / Take Profit ou le moment de l'expiration ont y changé), et cet ordre peut encore recevoir les données. Pour la réception garantie des données nouvelles sur l'ordre il est recommandé d'appeler la fonction OrderSelect() avant de les appeler directement.

Voir aussi

OrderGetInteger(), OrderGetDouble(), OrderGetString(), OrderCalcProfit(), OrderGetTicket(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

OrderGetDouble

Rend la propriété demandée de l'ordre préalablement choisi à l'aide de la fonction OrderGetTicket ou OrderSelect. La propriété de l'ordre doit être du type double. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
double  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id        // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,      // identificateur de la propriété
   double&                     double_var        // acceptons ici la valeur de la prop
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété d'ordre. La valeur peut être une des valeurs de l'énumération ENUM_ORDER_PROPERTY_DOUBLE.

double_var

[out] La valeur du type double, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type double.

Note

Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments" du terminal de client. Pour chaque symbole à tout moment on ne peut ouvrir qu'une position , tandis qu'on peut avoir plusieurs ordres remis selon le même symbole.

Pour la réception garantie des nouvelles données sur l'ordre il est recommandé d'appeler la fonction OrderSelect () avant de les appeler directement.

Voir aussi

OrdersTotal(), OrderGetTicket(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

OrderGetInteger

Rend la propriété demandée de l'ordre préalablement choisi à l'aide de la fonction OrderGetTicket ou OrderSelect. La propriété de l'ordre doit être du type datetime, int. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
long  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id,      // identificateur de la propriété
   long&                        long_var          // acceptons ici la valeur de la pro
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété d'ordre. La valeur peut être une des valeurs de l'énumération ENUM_ORDER_PROPERTY_INTEGER.

long_var

[out] La valeur du type long, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type long.

Note

Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments" du terminal de client. Pour chaque symbole à tout moment on ne peut ouvrir qu'une position , tandis qu'on peut avoir plusieurs ordres remis selon le même symbole.

Pour la réception garantie des nouvelles données sur l'ordre il est recommandé d'appeler la fonction OrderSelect () avant de les appeler directement.

Voir aussi

OrdersTotal(), OrderGetTicket(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

OrderGetString

Rend la propriété demandée de l'ordre préalablement choisi à l'aide de la fonction OrderGetTicket ou OrderSelect. La propriété de l'ordre doit être du type string. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
string  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id        // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id,      // identificateur de la propriété
   string&                     string_var        // acceptons ici la valeur de la prop
   );
```

Paramètres

property_id

[in] L'identificateur de la propriété d'ordre. La valeur peut être une des valeurs de l'énumération ENUM_ORDER_PROPERTY_STRING.

string_var

[out] La valeur du type strng, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type string.

Note

Il ne faut pas confondre les ordres remis et les positions entre eux-mêmes, qui s'affichent sur l'onglet "Commerce" au panneau "Instruments" du terminal de client. Pour chaque symbole à tout moment on ne peut ouvrir qu'une position , tandis qu'on peut avoir plusieurs ordres remis selon le même symbole.

Pour la réception garantie des nouvelles données sur l'ordre il est recommandé d'appeler la fonction OrderSelect () avant de les appeler directement.

Voir aussi

OrdersTotal(), OrderGetTicket(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

HistorySelect

Demande l'histoire des marchés et des ordres pour la période indiquée du temps de serveur.

```
bool  HistorySelect(
   datetime  from_date,     // de la date
   datetime  to_date        // jusqu'à la date
   );
```

Paramètres

from_date

[in] La date initiale de la demande.

to_date

[in] La date finale de la demande.

La valeur rendue

Rend true en cas du succès, autrement rend false.

Note

La fonction HistorySelect() crée dans le programme mql5 la liste des ordres et la liste des marchés pour l'appel ultérieur aux éléments de la liste par les fonctions correspondantes. On peut savoir la taille de la liste des marchés à l'aide de la fonction HistoryDealsTotal(), on peut recevoir la taille de la liste des ordres dans l'histoire par HistoryOrdersTotal(). C'est mieux de faire le triage des éléments de la liste des ordres par la fonction HistoryOrderGetTicket(), pour les éléments de la liste des marchés convient la fonction HistoryDealGetTicket().

Après l'application de la fonction HistoryOrderSelect() la liste des ordres dans l'histoire, accessible au programme mql5, est remise à zéro et se remplit de nouveau par l'ordre trouvé, si la recherche de l'ordre selon le ticket a terminé avec succès. Le même se rapporte à la liste des marchés accessibles au programme mql5- elle est remise à zéro par la fonction HistoryDealSelect() et se remplit de nouveau en cas de la réception réussie du marché selon le numéro du ticket.

Exemple:

```
void OnStart()
  {
   color BuyColor =clrBlue;
   color SellColor=clrRed;
//--- request trade history
   HistorySelect(0,TimeCurrent());
//--- create objects
   string   name;
   uint     total=HistoryDealsTotal();
   ulong    ticket=0;
   double   price;
   double   profit;
   datetime time;
   string   symbol;
   long     type;
```

© 2000-2015, MetaQuotes Software Corp.

```
   long     entry;
//--- for all deals
   for(uint i=0;i<total;i++)
     {
      //--- try to get deals ticket
      if(ticket=HistoryDealGetTicket(i))
        {
         //--- get deals properties
         price =HistoryDealGetDouble(ticket,DEAL_PRICE);
         time  =HistoryDealGetInteger(ticket,DEAL_TIME);
         symbol=HistoryDealGetString(ticket,DEAL_SYMBOL);
         type  =HistoryDealGetInteger(ticket,DEAL_TYPE);
         entry =HistoryDealGetInteger(ticket,DEAL_ENTRY);
         profit=HistoryDealGetDouble(ticket,DEAL_PROFIT);
         //--- only for current symbol
         if(price && time && symbol==Symbol())
           {
            //--- create price object
            name="TradeHistory_Deal_"+string(ticket);
            if(entry) ObjectCreate(0,name,OBJ_ARROW_RIGHT_PRICE,0,time,price,0,0);
            else      ObjectCreate(0,name,OBJ_ARROW_LEFT_PRICE,0,time,price,0,0);
            //--- set object properties
            ObjectSetInteger(0,name,OBJPROP_SELECTABLE,0);
            ObjectSetInteger(0,name,OBJPROP_BACK,0);
            ObjectSetInteger(0,name,OBJPROP_COLOR,type?BuyColor:SellColor);
            if(profit!=0) ObjectSetString(0,name,OBJPROP_TEXT,"Profit: "+string(profit
           }
        }
     }
//--- apply on chart
   ChartRedraw();
  }
```

Voir aussi

HistoryOrderSelect(), HistoryDealSelect()

© 2000-2015, MetaQuotes Software Corp.

HistorySelectByPosition

Demande l'histoire des marchés et les ordres avec l'indicateurindiqué de la position

```
bool  HistorySelectByPosition(
   long   position_id     // identificateur de la position - POSITION_IDENTIFIER
   );
```

Les paramètres

position_id

[in] L'identificateur de la position, qui est mis sur chaque ordre exécuté et sur chaque marché.

La valeur rendue

Rend true en cas du succès, autrement rend false.

Note

Il ne faut pas confondre les ordres de l'histoire commerciale et les ordres actifs remis entre eux-mêmes, qui sont affichés sur l'onglet "Commerce" au panneau "Instruments". On peut regarder la liste des ordres , qui étaient supprimés ou ont amené à la transaction commerciale, sur l'onglet "Histoire" au panneau "Instruments" du terminal de client.

La fonction HistorySelectByPosition() crée dans le programme mql5 la liste des ordres et la liste des marchés avec l'indicateur de la positionindiqué pour l'appel ultérieur aux éléments de la liste par les fonctions correspondantes. On peut savoir la taille de la liste des marchés à l'aide de la fonction HistoryDealsTotal(), on peut recevoir la taille de la liste des ordres dans l'histoire par HistoryOrdersTotal(). C'est mieux de faire le triage des éléments de la liste des ordres par la fonction HistoryOrderGetTicket(), pour les éléments de la liste des marchés convient la fonction HistoryDealGetTicket().

Après l'application de la fonction HistoryOrderSelect() la liste des ordres dans l'histoire, accessible au programme mql5, est remise à zéro et se remplit de nouveau par l'ordre trouvé, si la recherche de l'ordre selon le ticket a terminé avec succès. Le même se rapporte à la liste des marchés accessibles au programme mql5- elle est remise à zéro par la fonction HistoryDealSelect() et se remplit de nouveau en cas de la réception réussie du marché selon le numéro du ticket.

Voir aussi

HistorySelect(), HistoryOrderGetTicket(), Order Properties

© 2000-2015, MetaQuotes Software Corp.

HistoryOrderSelect

Choisit l'ordre dans l'histoire pour les références ultérieures à lui dans les fonctions correspondantes. Rend true à l'achèvement réussi de la fonction. Rend false à l'achèvement raté de la fonction. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

```
bool  HistoryOrderSelect(
   ulong  ticket,      // ticket de l'ordre
   );
```

Paramètres

ticket

[in] Le ticket de l'ordre

La valeur rendue

Rend true en cas du succès, autrement rend false.

Note

Il ne faut pas confondre les ordres de l'histoire commerciale et les ordres actifs remis entre eux-mêmes, qui sont affichés sur l'onglet "Commerce" au panneau "Instruments". On peut regarder la liste des ordres, qui étaient supprimés ou ont amené à la transaction commerciale, sur l'onglet "Histoire" au panneau "Instruments" du terminal de client.

La fonction HistoryOrderSelect() vide dans le programme mql5 la liste des ordres accessibles pour les appels, et y copie l'ordre unique, si l'exécution de HistoryOrderSelect() s'est achevée avec succès. S'il est nécessaire de trier tous les marchés choisis par la fonction HistorySelect(), il vaut mieux utiliser la fonction HistoryOrderGetTicket().

Voir aussi

HistorySelect(), HistoryOrderGetTicket(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

HistoryOrdersTotal

Rend le nombre d'ordres dans l'histoire. Avant d'appeler la fonction HistoryOrdersTotal() il est nécessaire de recevoir l'histoire des marchés et des ordres à l'aide de la fonction HistorySelect() ou HistorySelectByPosition().

```
int  HistoryOrdersTotal();
```

La valeur rendue

La valeur du type int.

Note

Il ne faut pas confondre les ordres de l'histoire commerciale et les ordres actifs remis entre eux-mêmes, qui sont affichés sur l'onglet "Commerce" au panneau "Instruments". On peut regarder la liste des ordres , qui étaient supprimés ou ont amené à la transaction commerciale, sur l'onglet "Histoire" au panneau "Instruments" du terminal de client.

Voir aussi

HistorySelect(), HistoryOrderSelect(), HistoryOrderGetTicket(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

HistoryOrderGetTicket

Rend le ticket de l'ordre correspondant dans l'histoire. Avant d'appeler la fonction HistoryOrderGetTicket() il est nécessaire de recevoir l'histoire des marchés et des ordres à l'aide de la fonctionHistorySelect() ouHistorySelectByPosition().

```
ulong  HistoryOrderGetTicket(
   int  index      // numéro dans la liste des ordres
   );
```

Paramètres

index

[in] Le numéro de l'ordre dans la liste des ordres.

La valeur rendue

La valeur du type ulong. En cas d'une mauvaise exécution rend 0.

Note

Il ne faut pas confondre les ordres de l'histoire commerciale et les ordres actifs remis entre eux-mêmes, qui sont affichés sur l'onglet "Commerce" au panneau "Instruments". On peut regarder la liste des ordres , qui étaient supprimés ou ont amené à la transaction commerciale, sur l'onglet "Histoire" au panneau "Instruments" du terminal de client.

Exemple:

```
void OnStart()
  {
   color BuyColor =clrBlue;
   color SellColor=clrRed;
//--- request trade history
   HistorySelect(0,TimeCurrent());
//--- create objects
   string   name;
   uint     total=HistoryDealsTotal();
   ulong    ticket=0;
   double   price;
   double   profit;
   datetime time;
   string   symbol;
   long     type;
   long     entry;
//--- for all deals
   for(uint i=0;i<total;i++)
     {
      //--- try to get deals ticket
      if((ticket=HistoryDealGetTicket(i))>0)
        {
         //--- get deals properties
         price =HistoryDealGetDouble(ticket,DEAL_PRICE);
```

© 2000-2015, MetaQuotes Software Corp.

```
         time  =(datetime)HistoryDealGetInteger(ticket,DEAL_TIME);
         symbol=HistoryDealGetString(ticket,DEAL_SYMBOL);
         type  =HistoryDealGetInteger(ticket,DEAL_TYPE);
         entry =HistoryDealGetInteger(ticket,DEAL_ENTRY);
         profit=HistoryDealGetDouble(ticket,DEAL_PROFIT);
         //--- only for current symbol
         if(price && time && symbol==Symbol())
           {
            //--- create price object
            name="TradeHistory_Deal_"+string(ticket);
            if(entry) ObjectCreate(0,name,OBJ_ARROW_RIGHT_PRICE,0,time,price,0,0);
            else      ObjectCreate(0,name,OBJ_ARROW_LEFT_PRICE,0,time,price,0,0);
            //--- set object properties
            ObjectSetInteger(0,name,OBJPROP_SELECTABLE,0);
            ObjectSetInteger(0,name,OBJPROP_BACK,0);
            ObjectSetInteger(0,name,OBJPROP_COLOR,type?BuyColor:SellColor);
            if(profit!=0) ObjectSetString(0,name,OBJPROP_TEXT,"Profit: "+string(profit
           }
        }
     }
//--- apply on chart
   ChartRedraw();
  }
```

Voir aussi

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

HistoryOrderGetDouble

Rend la propriété demandée de l'ordre dans l'histoire. La propriété de l'ordre doit être du type double. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
double  HistoryOrderGetDouble(
   ulong                      ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_DOUBLE  property_id       // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  HistoryOrderGetDouble(
   ulong                      ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,       // identificateur de la propriété
   double&                    double_var         // acceptons ici la valeur de la pro
   );
```

Paramètres

ticket_number

[in] Le ticket de l'ordre.

property_id

[in] L'identificateur de la propriété de l'ordre. La valeur peut être une des valeurs de l'énumération ENUM_ORDER_PROPERTY_DOUBLE.

double_var

[out] La variable du type double, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type double.

Note

Il ne faut pas confondre les ordres de l'histoire commerciale et les ordres actifs remis entre eux-mêmes, qui sont affichés sur l'onglet "Commerce" au panneau "Instruments". On peut regarder la liste des ordres, qui étaient supprimés ou ont amené à la transaction commerciale, sur l'onglet "Histoire" au panneau "Instruments" du terminal de client.

Voir aussi

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

HistoryOrderGetInteger

Rend la propriété demandée de l'ordre dans l'histoire. La propriété de l'ordre doit être du type datetime, int. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
long  HistoryOrderGetInteger(
   ulong                        ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  HistoryOrderGetInteger(
   ulong                        ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_INTEGER  property_id,       // identificateur de la propriété
   long&                        long_var           // acceptons ici la valeur de la pr
   );
```

Paramètres

ticket_number

[in] Le ticket de l'ordre.

property_id

[in] L'identificateur de la propriété de l'ordre. La valeur peut être une des valeurs de l'énumération ENUM_ORDER_PROPERTY_INTEGER.

long_var

[out] La variable du type long, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type long.

Note

Il ne faut pas confondre les ordres de l'histoire commerciale et les ordres actifs remis entre eux-mêmes, qui sont affichés sur l'onglet "Commerce" au panneau "Instruments". On peut regarder la liste des ordres, qui étaient supprimés ou ont amené à la transaction commerciale, sur l'onglet "Histoire" au panneau "Instruments" du terminal de client.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- recevrons le ticket du dernier ordre de l'histoire du commerce pour une semaine
   ulong last_order=GetLastOrderTicket();
   if(HistoryOrderSelect(last_order))
     {
      //--- le temps de l'organisation de l'ordre en millisecondes de 01.01.1970
      long time_setup_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_SETUP_MSC);
      PrintFormat("Order #%d ORDER_TIME_SETUP_MSC=%i64 => %s",
                  last_order,time_setup_msc,TimeToString(time_setup_msc/1000));
      //--- le temps de l'exécution/la suppression de l'ordre en 01.01.1970
      long  time_done_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_DONE_MSC);
      PrintFormat("Order #%d ORDER_TIME_DONE_MSC=%i64 => %s",
                  last_order,time_done_msc,TimeToString(time_done_msc/1000));
     }
   else // informons de l'échec
      PrintFormat("HistoryOrderSelect() failed for #%d. Eror code=%d",
                  last_order,GetLastError());

//---
  }
//+------------------------------------------------------------------+
//| rend le ticket du dernier ordre dans l'histoire ou -1            |
//+------------------------------------------------------------------+
ulong GetLastOrderTicket()
  {
//---demanderons l'histoire pour les derniers 7 jours
   if(!GetTradeHistory(7))
     {
      //--- informons de l'appel non réussi et rendons -1
      Print(__FUNCTION__," HistorySelect() a rendu false");
      return -1;
     }
//---
   ulong first_order,last_order,orders=HistoryOrdersTotal();
//--- s'il y a des ordres, commençons à travailler avec eux
   if(orders>0)
     {
      Print("Orders = ",orders);
      first_order=HistoryOrderGetTicket(0);
      PrintFormat("first_order = %d",first_order);
      if(orders>1)
        {
         last_order=HistoryOrderGetTicket((int)orders-1);
         PrintFormat("last_order = %d",last_order);
         return last_order;
        }
      return first_order;
     }
//--- n'ont pas trouvé aucun ordre, rendons -1
   return -1;
  }
//+------------------------------------------------------------------+
//| demande l'histoire pour les derniers jours et rendra false à l'échec  |
//+------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- spécifions le temps de la période d'une semaine pour la demande de l'histoire co
```

© 2000-2015, MetaQuotes Software Corp.

```
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//--- faisons la demande et vérifions le résultat
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
      return false;
     }
//--- l'histoire est reçue avec succès
   return true;
  }
```

Voir aussi

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

HistoryOrderGetString

Rend la propriété demandée de l'ordre dans l'histoire. La propriété de l'ordre doit être du type string. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
string  HistoryOrderGetString(
   ulong                      ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_STRING  property_id       // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  HistoryOrderGetString(
   ulong                      ticket_number,     // ticket
   ENUM_ORDER_PROPERTY_STRING  property_id,      // identificateur de la propriété
   string&                    string_var         // acceptons ici la valeur de la pro
   );
```

Paramètres

ticket_number

[in] Le ticket de l'ordre.

property_id

[in] L'identificateur de la propriété de l'ordre. La valeur peut être une des valeurs de l'énumération ENUM_ORDER_PROPERTY_STRING.

string_var

[out] La variable du type strng, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type string.

Note

Il ne faut pas confondre les ordres de l'histoire commerciale et les ordres actifs remis entre eux-mêmes, qui sont affichés sur l'onglet "Commerce" au panneau "Instruments". On peut regarder la liste des ordres, qui étaient supprimés ou ont amené à la transaction commerciale, sur l'onglet "Histoire" au panneau "Instruments" du terminal de client.

Voir aussi

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriétés des ordres

© 2000-2015, MetaQuotes Software Corp.

HistoryDealSelect

Choisit le marché dans l'histoire pour davantage l'appeler par les fonctions appropriées. Rend true à l'achèvement réussi de la fonction. Rend false à l'achèvement raté de la fonction. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

```
bool  HistoryDealSelect(
   ulong  ticket,     // ticket du marché
   );
```

Paramètres

ticket

[in] Le ticket du marché

La valeur rendue

Rend true en cas du succès, autrement rend false.

Note

Il ne faut pas confondre entre eux-mêmesles ordres, les marchés et les positions. Chaque marché est le résultat de l'exécution d'un certain ordre, chaque position est le résultat final d'un ou quelques marchés.

La fonction HistoryDealSelect () vide dans le programme mql5 la liste des marchés accessibles pour les appels, et y copie le marché unique, si l'exécution de HistoryDealSelect () s'est achevée avec succès. S'il est nécessaire de trier tous les marchés choisis par la fonction HistorySelect () , il vaut mieux utiliser la fonction HistoryDealGetTicket().

Voir aussi

HistorySelect(), HistoryDealGetTicket(), Propriété des marchés

© 2000-2015, MetaQuotes Software Corp.

HistoryDealsTotal

Rend le nombre de marchés dans l'histoire. Avant d'appeler la fonction HistoryDealsTotal() il est nécessaire de recevoir l'histoire des marchés et des ordres à l'aide de la fonction HistorySelect() ou HistorySelectByPosition().

```
int  HistoryDealsTotal();
```

La valeur rendue

La valeur du type int.

Note

Il ne faut pas confondre entre eux-mêmesles ordres, les marchés et les positions. Chaque marché est le résultat de l'exécution d'un certain ordre, chaque position est le résultat final d'un ou quelques marchés.

Voir aussi

HistorySelect(), HistoryDealGetTicket(), Propriété des marchés

© 2000-2015, MetaQuotes Software Corp.

HistoryDealGetTicket

Choisit le marché pour le traitement ultérieur et rend le ticket les marchés dans l'histoire. Avant d'appeler la fonction HistoryDealGetTicket() il est nécessaire de recevoir l'histoire des marchés et des ordres à l'aide de la fonction HistorySelect() ou HistorySelectByPosition().

```
ulong  HistoryDealGetTicket(
   int  index      // numéro du marché
   );
```

Paramètres

index

[in] Le numéro du marché dans la liste des marchés.

La valeur rendue

La valeur du type ulong. En cas d'une mauvaise exécution rend 0.

Note

Il ne faut pas confondre entre eux-mêmesles ordres, les marchés et les positions. Chaque marché est le résultat de l'exécution d'un certain ordre, chaque position est le résultat final d'un ou quelques marchés.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
void OnStart()
  {
   ulong deal_ticket;            // ticket de marché
   ulong order_ticket;           // ticket d'ordre, selon qui on faisait le marché
   datetime transaction_time;    // temps de l'accomplissement du marché
   long deal_type ;              // type de l'opération commerciale
   long position_ID;             // identificateur de la position
   string deal_description;      // description de l'opération
   double volume;                // volume de l'opération
   string symbol;                // selon quel symbole il y avait un marché
//--- établissons la date initiale et finale pour la requête de l'histoire des marchés
   datetime from_date=0;          // dès le début
   datetime to_date=TimeCurrent();// jusqu'à la situation actuelle
//--- demanderons l'histoire des marchés dans l'intervalle indiqué
   HistorySelect(from_date,to_date);
//--- le total dans le répertoire des marchés
   int deals=HistoryDealsTotal();
//--- maintenant nous traitons chaque marché
   for(int i=0;i<deals;i++)
     {
      deal_ticket=               HistoryDealGetTicket(i);
      volume=                    HistoryDealGetDouble(deal_ticket,DEAL_VOLUME);
      transaction_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
      order_ticket=              HistoryDealGetInteger(deal_ticket,DEAL_ORDER);
      deal_type=                 HistoryDealGetInteger(deal_ticket,DEAL_TYPE);
      symbol=                    HistoryDealGetString(deal_ticket,DEAL_SYMBOL);
      position_ID=               HistoryDealGetInteger(deal_ticket,DEAL_POSITION_ID);
      deal_description=          GetDealDescription(deal_type,volume,symbol,order_tick
      //---  faisons un beau formatage pour le numéro du marché
      string print_index=StringFormat("% 3d",i);
      //--- sortons l'information pour le marché
      Print(print_index+": deal #",deal_ticket," at ",transaction_time,deal_descriptio
     }
  }
//+------------------------------------------------------------------+
//| Rend la description de chaîne de l'opération                     |
//+------------------------------------------------------------------+
string GetDealDescription(long deal_type,double volume,string symbol,long ticket,long
  {
   string descr;
//---
   switch(deal_type)
     {
      case DEAL_TYPE_BALANCE:            return ("balance");
      case DEAL_TYPE_CREDIT:             return ("credit");
      case DEAL_TYPE_CHARGE:             return ("charge");
      case DEAL_TYPE_CORRECTION:         return ("correction");
      case DEAL_TYPE_BUY:                descr="buy"; break;
      case DEAL_TYPE_SELL:               descr="sell"; break;
      case DEAL_TYPE_BONUS:              return ("bonus");
      case DEAL_TYPE_COMMISSION:         return ("additional commission");
      case DEAL_TYPE_COMMISSION_DAILY:   return ("daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY: return ("monthly commission");
      case DEAL_TYPE_AGENT_DAILY:        return ("daily agent commission");
      case DEAL_TYPE_AGENT_MONTHLY:      return ("monthly agent commission");
      case DEAL_TYPE_INTERESTRATE:       return ("interest rate");
      case DEAL_TYPE_BUY_CANCELED:       descr="cancelled buy deal"; break;
      case DEAL_TYPE_SELL_CANCELED:      descr="cancelled sell deal"; break;
     }
   descr=StringFormat("%s %G %s (order #%d, position ID %d)",
                      descr,  // description courante
```

© 2000-2015, MetaQuotes Software Corp.

```
                        volume, // volume du marché
                        symbol, // instrument du marché
                        ticket, // ticket de l'ordre qui a provoqué le marché
                        pos_ID  // ID de la position, à qui participait le marché
                        );
   return(descr);
//---
  }
```

Voir aussi

HistorySelect(), HistoryDealsTotal(), HistoryDealSelect(), Propriété des marchés

© 2000-2015, MetaQuotes Software Corp.

HistoryDealGetDouble

Rend la propriété demandée du marché. La propriété du marché doit être du type double. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
double  HistoryDealGetDouble(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_DOUBLE  property_id        // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  HistoryDealGetDouble(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_DOUBLE  property_id,       // identificateur de la propriété
   double&                    double_var         // acceptons ici la valeur de la prop
   );
```

Paramètres

ticket_number

[in] Le ticket de l'ordre.

property_id

[in] L'identificateur de la propriété de l'ordre. La valeur peut être une des valeurs de l'énumération ENUM_DEAL_PROPERTY_DOUBLE.

double_var

[out] La variable du type double, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type double.

Note

Il ne faut pas confondre entre eux-mêmes les ordres, les marchés et les positions. Chaque marché est le résultat de l'exécution d'un certain ordre, chaque position est le résultat final d'un ou quelques marchés.

Voir aussi

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Propriété des marchés

© 2000-2015, MetaQuotes Software Corp.

HistoryDealGetInteger

Rend la propriété demandée de l'ordre dans l'histoire. La propriété de l'ordre doit être du type datetime, int. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
long  HistoryDealGetInteger(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_INTEGER  property_id       // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  HistoryDealGetInteger(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_INTEGER  property_id,       // identificateur de la propriété
   long&                      long_var           // acceptons ici la valeur de la pro
   );
```

Paramètres

ticket_number

[in] Le ticket de l'ordre.

property_id

[in] L'identificateur de la propriété de l'ordre. La valeur peut être une des valeurs de l'énumération ENUM_DEAL_PROPERTY_INTEGER.

long_var

[out] La variable du type long,acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type long.

Note

Il ne faut pas confondre entre eux-mêmes les ordres, les marchés et les positions. Chaque marché est le résultat de l'exécution d'un certain ordre, chaque position est le résultat final d'un ou quelques marchés.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- recevrons le ticket de la dernière transaction de l'histoire du commerce pour ur
   ulong last_deal=GetLastDealTicket();
   if(HistoryDealSelect(last_deal))
     {
      //--- l'heure de la transaction en millisecondes à partir du 01.01.1970
      long deal_time_msc=HistoryDealGetInteger(last_deal,DEAL_TIME_MSC);
      PrintFormat("Deal #%d DEAL_TIME_MSC=%i64 => %s",
                  last_deal,deal_time_msc,TimeToString(deal_time_msc/1000));
     }
   else
      PrintFormat("HistoryDealSelect() failed for #%d. Eror code=%d",
                  last_deal,GetLastError());
//---
  }
//+------------------------------------------------------------------+
//| Rend le ticket de la dernière transaction dans l'histoire ou -1  |
//+------------------------------------------------------------------+
ulong GetLastDealTicket()
  {
//--- demanderons l'histoire pour les derniers 7 jours
   if(!GetTradeHistory(7))
     {
      //--- informons de l'appel non réussi et rendons -1
      Print(__FUNCTION__," HistorySelect() a rendu false");
      return -1;
     }
//---
   ulong first_deal,last_deal,deals=HistoryOrdersTotal();
//--- s'il y a des ordres, commençons à travailler avec eux
   if(deals>0)
     {
      Print("Deals = ",deals);
      first_deal=HistoryDealGetTicket(0);
      PrintFormat("first_deal = %d",first_deal);
      if(deals>1)
        {
         last_deal=HistoryDealGetTicket((int)deals-1);
         PrintFormat("last_deal = %d",last_deal);
         return last_deal;
        }
      return first_deal;
     }
//--- on n'a pas trouvé aucune transaction, rendons -1
   return -1;
  }
//+------------------------------------------------------------------+
//| demande l'histoire pour les derniers jours et rendra false à l'échec |
//+------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- spécifions la période d'une semaine du temps pour la demande de l'histoire comme
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//---faisons la demande et vérifions le résultat
   if(!HistorySelect(from,to))
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
       return false;
      }
//--- l'histoire est reçue avec succès
   return true;
  }
```

Voir aussi

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Propriété des marchés

© 2000-2015, MetaQuotes Software Corp.

HistoryDealGetString

Rend la propriété demandée de l'ordre dans l'histoire. La propriété de l'ordre doit être du type string. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
string  HistoryDealGetString(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_STRING  property_id        // identificateur de la propriété
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  HistoryDealGetString(
   ulong                      ticket_number,     // ticket
   ENUM_DEAL_PROPERTY_STRING  property_id,       // identificateur de la propriété
   string&                    string_var         // acceptons ici la valeur de la prop
   );
```

Paramètres

ticket_number

[in] Le ticket de l'ordre.

property_id

[in] L'identificateur de la propriété de l'ordre. La valeur peut être une des valeurs de l'énumération ENUM_DEAL_PROPERTY_STRING.

string_var

[out] La variable du type string, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type string.

Note

Il ne faut pas confondre entre eux-mêmes les ordres, les marchés et les positions. Chaque marché est le résultat de l'exécution d'un certain ordre, chaque position est le résultat final d'un ou quelques marchés.

Voir aussi

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Propriété des marchés

© 2000-2015, MetaQuotes Software Corp.

Управление сигналами

Группа функций, предназначенных для управления торговыми сигналами. Данные функции позволяют:

- получать информацию о торговых сигналах, доступных для копирования,
- прочитать или установить настройки копирования торговых сигналов,
- произвести подписку на сигнал и ее отмену средствами языка MQL5.

Функция	Действие
SignalBaseGetDouble	Возвращает значение свойства типа double для выбранного сигнала
SignalBaseGetInteger	Возвращает значение свойства типа integer для выбранного сигнала
SignalBaseGetString	Возвращает значение свойства типа string для выбранного сигнала
SignalBaseSelect	Выбирает для работы сигнал из базы торговых сигналов, доступных в терминале
SignalBaseTotal	Возвращает общее количество сигналов, доступных в терминале
SignalInfoGetDouble	Возвращает из настроек копирования торгового сигнала значение свойства типа double
SignalInfoGetInteger	Возвращает из настроек копирования торгового сигнала значение свойства типа integer
SignalInfoGetString	Возвращает из настроек копирования торгового сигнала значение свойства типа string
SignalInfoSetDouble	Устанавливает в настройках копирования торгового сигнала значение свойства типа double
SignalInfoSetInteger	Устанавливает в настройках копирования торгового сигнала значение свойства типа integer
SignalSubscribe	Производит подписку на копирование торгового сигнала
SignalUnsubscribe	Отменяет подписку на копирование торгового сигнала

© 2000-2015, MetaQuotes Software Corp.

SignalBaseGetDouble

Возвращает значение свойства типа double для выбранного сигнала.

```
double  SignalBaseGetDouble(
   ENUM_SIGNAL_BASE_DOUBLE     property_id,     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_BASE_DOUBLE.

Возвращаемое значение

Значение типа double указанного свойства сигнала.

© 2000-2015, MetaQuotes Software Corp.

SignalBaseGetInteger

Возвращает значение свойства типа integer для выбранного сигнала.

```
long  SignalBaseGetInteger(
   ENUM_SIGNAL_BASE_INTEGER     property_id,     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_BASE_INTEGER.

Возвращаемое значение

Значение типа integer указанного свойства сигнала.

© 2000-2015, MetaQuotes Software Corp.

SignalBaseGetString

Возвращает значение свойства типа string для выбранного сигнала.

```
string  SignalBaseGetString(
   ENUM_SIGNAL_BASE_STRING     property_id,     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_BASE_STRING.

Возвращаемое значение

Значение типа string указанного свойства сигнала.

© 2000-2015, MetaQuotes Software Corp.

SignalBaseSelect

Выбирает для работы сигнал из базы торговых сигналов, доступных в терминале.

```
bool  SignalBaseSelect(
   int      index      // индекс записи сигнала
   );
```

Параметры

index

[in] Индекс записи сигнала в базе торговых сигналов.

Возвращаемое значение

Возвращает true при успешном завершении функции или false в случае ошибки. Чтобы получить информацию об ошибке, необходимо вызвать функцию GetLastError().

Пример:

```
void OnStart()
  {
//--- запрашиваем общее количество сигналов в базе
   int total=SignalBaseTotal();
//--- цикл по всем сигналам
   for(int i=0;i<total;i++)
     {
      //--- выбираем сигнал для дальнейшей работы
      if(SignalBaseSelect(i))
        {
         //--- получение свойств сигнала
         long   id    =SignalBaseGetInteger(SIGNAL_BASE_ID);           // id сигнала
         long   pips  =SignalBaseGetInteger(SIGNAL_BASE_PIPS);         // результат тор
         long   subscr=SignalBaseGetInteger(SIGNAL_BASE_SUBSCRIBERS); // количество по
         string name  =SignalBaseGetString(SIGNAL_BASE_NAME);          // имя сигнала
         double price =SignalBaseGetDouble(SIGNAL_BASE_PRICE);         // цена подписки
         string curr  =SignalBaseGetString(SIGNAL_BASE_CURRENCY);      // валюта сигнал
         //--- выводим все прибыльные бесплатные сигналы с ненулевым количеством подпи
         if(price==0.0 && pips>0 && subscr>0)
            PrintFormat("id=%d, name=\"%s\", currency=%s, pips=%d, subscribers=%d",id,
        }
      else PrintFormat("Ошибка выбора сигнала. Код ошибки=%d",GetLastError());
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

SignalBaseTotal

Возвращает общее количество сигналов, доступных в терминале.

```
int  SignalBaseTotal();
```

Возвращаемое значение

Общее количество сигналов, доступных в терминале.

© 2000-2015, MetaQuotes Software Corp.

SignalInfoGetDouble

Возвращает из настроек копирования торгового сигнала значение свойства типа double.

```
double  SignalInfoGetDouble(
   ENUM_SIGNAL_INFO_DOUBLE     property_id,     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_INFO_DOUBLE.

Возвращаемое значение

Значение типа double указанного свойства настроек копирования торгового сигнала.

© 2000-2015, MetaQuotes Software Corp.

SignalInfoGetInteger

Возвращает из настроек копирования торгового сигнала значение свойства типа integer.

```
long  SignalInfoGetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_INFO_INTEGER.

Возвращаемое значение

Значение типа integer указанного свойства настроек копирования торгового сигнала.

© 2000-2015, MetaQuotes Software Corp.

SignalInfoGetString

Возвращает из настроек копирования торгового сигнала значение свойства типа string.

```
string  SignalInfoGetString(
   ENUM_SIGNAL_INFO_STRING     property_id,     // идентификатор свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства настроек копирования торгового сигнала. Значение может быть одним из значений перечисления ENUM_SIGNAL_INFO_STRING.

Возвращаемое значение

Значение типа string указанного свойства настроек копирования торгового сигнала.

© 2000-2015, MetaQuotes Software Corp.

SignalInfoSetDouble

Устанавливает в настройках копирования торгового сигнала значение свойства типа double.

```
bool  SignalInfoSetDouble(
   ENUM_SIGNAL_INFO_DOUBLE      property_id,     // идентификатор свойства
   double                       value            // значение свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_INFO_DOUBLE.

value

[in] Значение свойства настроек копирования торгового сигнала.

Возвращаемое значение

Возвращает true в случае успешного изменения свойства, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2015, MetaQuotes Software Corp.

SignalInfoSetInteger

Устанавливает в настройках копирования торгового сигнала значение свойства типа integer.

```
bool  SignalInfoSetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // идентификатор свойства
   long                         value            // значение свойства
   );
```

Параметры

property_id

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления ENUM_SIGNAL_INFO_INTEGER.

value

[in] Значение свойства настроек копирования торгового сигнала.

Возвращаемое значение

Возвращает true в случае успешного изменения свойства, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2015, MetaQuotes Software Corp.

SignalSubscribe

Производит подписку на копирование указанного торгового сигнала.

```
bool  SignalSubscribe(
   long     signal_id     // id сигнала
   );
```

Параметры

signal_id

[in] Идентификатор сигнала.

Возвращаемое значение

Возвращает true в случае успешной подписки на копирование торгового сигнала, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2015, MetaQuotes Software Corp.

SignalUnsubscribe

Отменяет подписку на копирование торгового сигнала.

```
bool  SignalUnsubscribe();
```

Возвращаемое значение

Возвращает true в случае успешной отмены подписки на копирование торгового сигнала, иначе возвращает false. Чтобы получить дополнительную информацию об ошибке, необходимо вызвать функцию GetLastError().

© 2000-2015, MetaQuotes Software Corp.

Les variables globales du terminal de client

Le groupe des fonctions destinées au travail avec les variables globales.

Il ne faut pas confondre les variables globales du terminal de client avec les variables déclarées au niveau global du programme mql5.

Les variables globales existent dans le terminal de client 4 semaines dès le moment du dernier appel, après cela elles sont supprimés automatiquement. Non seulement l'installation de la nouvelle valeur est considérée comme l'appel vers la variable globale, mais aussi la lecture de la valeur de la variable globale.

Les variables globales du terminal de client sont accessibles simultanément de tous les programmes mql5 lancés sur le terminal de client.

Fonction	Action
GlobalVariableCheck	Contrôle l'existence de la variable globale avec le nom indiqué
GlobalVariableTime	Rend le temps du dernier accès à la variable globale
GlobalVariableDel	Supprime la variable globale
GlobalVariableGet	Demande la valeur de la variable globale
GlobalVariableName	Rend le nom de la variable globale selon le numéro d'ordre dans le répertoire des variables globales
GlobalVariableSet	Met la nouvelle valeur à une variable globale
GlobalVariablesFlush	Inscrit forcément le contenu de toutes les variables globales sur le disque
GlobalVariableTemp	Met une nouvelle valeur de la variable globale, qui existe seulement pendant la session courante du terminal
GlobalVariableSetOnCondition	Met une nouvelle valeur de la variable existant globale selon la condition
GlobalVariablesDeleteAll	Supprime les variables globales avec le préfixe indiqué dans le nom
GlobalVariablesTotal	Rend le total des variables globales

© 2000-2015, MetaQuotes Software Corp.

GlobalVariableCheck

Contrôle l'existence de la variable globale avec le nom indiqué.

```
bool  GlobalVariableCheck(
   string  name      // nom
   );
```

Paramètres

name

[in] Le nom de la variable globale.

La valeur rendue

Rend la valeur true, si la variable globale existe, autrement rend false.

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

Voir aussi

GlobalVariableTime()

© 2000-2015, MetaQuotes Software Corp.

GlobalVariableTime

Rend le temps du dernier accès à la variable globale.

```
datetime  GlobalVariableTime(
   string  name      // nom
   );
```

Paramètres

name

[in] Le nom de la variable globale.

La valeur rendue

Rend le temps du dernier accès à la variable indiquée globale. L'appel à la variable pour la valeur est aussi l'accès. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

Voir aussi

GlobalVariableCheck()

© 2000-2015, MetaQuotes Software Corp.

GlobalVariableDel

Supprime la variable globale du terminal de client.

```
bool  GlobalVariableDel(
   string  name      // nom
   );
```

Paramètres

name

[in] Le nom de la variable globale.

La valeur rendue

A l'effacement réussi la fonction rend true, autrement rend false. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

© 2000-2015, MetaQuotes Software Corp.

GlobalVariableGet

Rend la valeur de la variable globale existante du terminal de client. Il y a 2 variantes de la fonction.

1. Rend directement la valeur de la propriété.

```
double  GlobalVariableGet(
   string  name      // nom
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la variable globale du terminal de client se place à la variable de réception transmise selon le lien par le deuxième paramètre.

```
bool  GlobalVariableGet(
   string  name               // nom
   double& double_var         // acceptons ici la valeur de la variable globale
   );
```

Paramètres

name

[in] Le nom de la variable globale.

double_var

[out] La variable du type double, acceptant la valeur qui se trouve dans la variable globale du terminal de client.

La valeur rendue

La valeur de la variable globale existante ou 0 en cas de l'erreur. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

© 2000-2015, MetaQuotes Software Corp.

GlobalVariableName

Rend le nom de la variable globale selon le numéro d'ordre.

```
string  GlobalVariableName(
   int  index      // numéro dans le répertoire des variables globales
   );
```

Paramètres

index

[in] Le numéro d'ordre dans le répertoire des variables globales. Doit être plus ou égal à 0 et moins que GlobalVariablesTotal().

La valeur rendue

Le nom de la variable globale selon le numéro d'ordre dans la liste des variables globales.Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

© 2000-2015, MetaQuotes Software Corp.

GlobalVariableSet

Met la nouvelle valeur à une variable globale. Si la variable n'existe pas, le système crée une nouvelle variable globale.

```
datetime  GlobalVariableSet(
   string  name,      // nom
   double  value      // valeur à mettre
   );
```

Paramètres

name

[in] Le nom de la variable globale.

value

[in] Une nouvelle valeur numérique.

La valeur rendue

À l'exécution réussie la fonction rend le temps du dernier accès, autrement c'est 0. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Имя графического объекта не должно превышать 63 символа. Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

© 2000-2015, MetaQuotes Software Corp.

GlobalVariablesFlush

Inscrit forcément le contenu de toutes les variables globales sur le disque.

```
void  GlobalVariablesFlush();
```

La valeur rendue

Il n'y a pas de valeur rendue.

Note

Le terminal inscrit lui-même toutes les variables globales à la fin du travail, mais à la panne soudaine du travail de l'ordinateur les données peuvent se perdre. Cette fonction permet de maîtriser indépendamment le procès de la sauvegarde des variables globales sur le cas des situations anormales.

© 2000-2015, MetaQuotes Software Corp.

GlobalVariableTemp

Produit la tentative de la création de la variable temporaire globale. Si la variable n'existe pas, le système crée une nouvelle variable temporaire globale.

```
bool  GlobalVariableTemp(
   string  name,      // nom
   );
```

Paramètres

name

[in] Le nom de la variable temporaire globale.

La valeur rendue

A l'exécution réussi la fonction rend true, autrement rend false. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Les variables temporaires globales existent seulement au temps de travail du terminal de client, après la clôture du terminal ils sont supprimés automatiquement. A l'exécution de l'opération GlobalVariablesFlush() les variables temporaires globales ne s'inscrivent pas sur le disque.

Après la création de la variable temporaire globale l'accès à cette variable et sa modification se réalise ainsi que vers la variable ordinaire globale du terminal de client.

© 2000-2015, MetaQuotes Software Corp.

GlobalVariableSetOnCondition

Met une nouvelle valeur de la variable globale existante si la valeur courante est égal à la valeur du troisième paramètre check_value. Si la variable n'existe pas, la fonction générera l'erreur ERR_GLOBALVARIABLE_NOT_FOUND (4501) et rendra false.

```
bool  GlobalVariableSetOnCondition(
   string  name,            // nom
   double  value,           // valeur à l'exécution de la condition
   double  check_value      // condition vérifiée
   );
```

Paramètres

name

[in] Le nom de la variable globale.

value

[in] Une nouvelle valeur.

check_value

[in] La valeur pour la vérification de la valeur courante de la variable globale.

La valeur rendue

A l'exécution réussi la fonction rend true, autrement rend false. Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError(). Si la valeur courante de la variable globale se distingue de check_value, la fonction rendra false.

Note

La fonction assure l'accès atomique à la variable globale, c'est pourquoi elle peut être utilisée pour l'organisation du mutex à la coopération de quelques experts travaillant simultanément dans la limite d'un terminal de client.

© 2000-2015, MetaQuotes Software Corp.

GlobalVariablesDeleteAll

Supprime les variables globales du terminal de client.

```
int  GlobalVariablesDeleteAll(
   string     prefix_name=NULL      // toutes les variables globales, dont les noms co
   datetime   limit_data=0          // toutes les variables globales, qui changeaient
   );
```

Paramètres

prefix_name=NULL

[in] Le préfixe du nom des variables globales supprimées. Si on indique le préfixe NULL ou la chaîne vide, toutes les variables globales conviennent au critère de l'effacement ,correspondantes au critère de l'effacement selon la date

limit_data=0

[in] La date pour la sélection des variables globales par le temps de la dernière modification. Les variables globales se suppriment, qui changeaient avant la date indiquée. Si le paramètre est égal au zéro, toutes les variables globales, correspondantes au premier critère (selon le préfixe) se suppriment.

La valeur rendue

Le nombre de variables supprimées.

Note

Si les deux paramètres sont égaux au zéro (prefix_name=NULL et limit_data=0),toutes les variables globales du terminal se suppriment. Si on indique les deux paramètres, se suppriment les variables globales correspondantes simultanément à chacun des paramètres indiqués.

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement.

© 2000-2015, MetaQuotes Software Corp.

GlobalVariablesTotal

Rend le total des variables globales du terminal de client.

```
int  GlobalVariablesTotal();
```

La valeur rendue

Le nombre de variables globales.

Note

Les variables globales existent dans le terminal de client les 4 semaines dès le moment du dernier appel, après cela elles se suppriment automatiquement. Non seulement l'installation de la nouvelle valeur est considérée comme l'appel à la variable globale, mais aussi la lecture de la valeur de la variable globale.

© 2000-2015, MetaQuotes Software Corp.

Les opérations de fichier

Le groupe des fonctions pour le travail avec les fichiers.

Il y a deux répertoires (avec les sous-directoires), dans lesquels les fichiers de travail peuvent être installés:

- le répertoire_des données_du terminal\MQL5\FILES\ (choisissez le point du menu "Fichier" - "Ouvrir le répertoire des données" pour voir dans le terminal);
- le dossier total de tous les terminaux établis sur l'ordinateur - est d'habitude disposé dans le répertoire C:\Documents and Settings\All Users\Application Data\MetaQuotes\Terminal\Common \Files.

D'une manière de programme on peut recevoir les noms de ces répertoires à l'aide de la fonction TerminalInfoString(), en utilisant les énumérations ENUM_TERMINAL_INFO_STRING:

```
//--- dossier, où se trouvent les données du terminal
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
//--- Dossier total de tous les terminaux de client
   string common_data_path=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
```

Le travail avec les fichiers d'autres répertoires est interdit.

Les fonctions de fichier permettent de travailler avec des soi-disant "canaux nommés". Pour cela il suffit d'appeler la fonctionFileOpen() avec des paramètres appropriés.

Fonction	Action
FileFindFirst	Commence l'excédent des fichiers dans le répertoire correspondant conformément au filtre indiqué
FileFindNext	Continue la recherche commencée par la fonction FileFindFirst()
FileFindClose	Ferme le handle de la recherche
FileOpen	Ouvre un fichier avec un nom et un drapeau indiqués
FileDelete	Supprime le fichier indiqué
FileFlush	Écrit à un disque toutes les données restant dans le tampon du fichier d'entrée-de sortie
FileGetInteger	Reçoit la propriété entière du fichier
FileIsEnding	Définit la fin du fichier en train de la lecture
FileIsLineEnding	Définit la fin de la ligne dans le fichier du texte en train de la lecture
FileClose	Ferme le fichier auparavant ouvert
FileIsExist	Vérifie l'existence d'un fichier
FileCopy	Copie le fichier original d'un répertoire local ou

© 2000-2015, MetaQuotes Software Corp.

	commun à un autre fichier
FileMove	Déplace et renomme le fichier
FileReadArray	Lit les tableaux de tous les types, sauf les tableaux de chaîne (peut être un tableau des structures qui ne contient pas les chaînes et les tableaux dynamiques), du fichier binaire de la position courante de l'indicateur de fichier
FileReadBool	Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type bool
FileReadDatetime	Lit du fichier du type CSV la chaîne d'un des formats: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" ou "HH:MI:SS" - et la transforme en valeur du type datetime
FileReadDouble	Lit le nombre de l'exactitude double avec une virgule flottante (double) du fichier binaire de la position courante de l'indicateur de fichier
FileReadFloat	Lit de la position courante de l'indicateur de fichier la valeur du type float
FileReadInteger	Lit du fichier binaire la valeur du type int, short ou char en fonction de la longueur indiquée dans les bytes
FileReadLong	Lit de la position courante de l'indicateur de fichier la valeur du type long
FileReadNumber	Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type double
FileReadString	Lit du fichier la chaîne de la position courante de l'indicateur de fichier
FileReadStruct	Lit du fichier binaire le contenu à la structure transmise à titre du paramètre
FileSeek	Déplace la position de l'indicateur de fichier par un nombre spécifié de bytes par rapport à la position indiquée
FileSize	Rend la dimension d'un fichier ouvert correspondant
FileTell	Rend la position courante de l'indicateur de fichier du fichier correspondant ouvert
FileWrite	Inscrit les données au fichier comme CSV ou TXT

© 2000-2015, MetaQuotes Software Corp.

FileWriteArray	Inscrit au fichier du type BIN les tableaux des tous les types, sauf les tableaux de chaîne
FileWriteDouble	Inscrit au fichier binaire la valeur du paramètre du type double de la position courante de l'indicateur de fichier
FileWriteFloat	Inscrit au fichier binaire la valeur du paramètre du type float de la position courante de l'indicateur de fichier
FileWriteInteger	Inscrit au fichier binaire la valeur du paramètre du type int de la position courante de l'indicateur de fichier
FileWriteLong	Inscrit au fichier binaire la valeur du paramètre du type long de la position courante de l'indicateur de fichier
FileWriteString	Inscrit au fichier du type BIN ou TXT la valeur du paramètre du type string de la position courante de l'indicateur de fichier
FileWriteStruct	Inscrit au fichier binaire le contenu de la structure, transmise à titre du paramètre de la position courante de l'indicateur de fichier
FolderCreate	Crée le répertoire dans le dossier Files (en fonction de la valeur common_flag)
FolderDelete	Supprime le répertoire indiqué. Si le dossier n'est pas vide, elle ne peut pas être supprimé
FolderClean	Supprime tous les fichiers dans le dossier indiqué

Si le fichier s'ouvre pour l'enregistrement à l'aide de la fonction FileOpen(), tous les sous-dossiers, indiqué au chemin seront créés en cas de leur absence.

© 2000-2015, MetaQuotes Software Corp.

FileFindFirst

Commence le balayage des fichiers et des sous-répertoires dans le directoire correspondant conformément au filtre indiqué.

```
long  FileFindFirst(
   const string   file_filter,          // chaîne - le filtre de la recherche
   string&        returned_filename,    // le nom du fichier trouvé ou du sous-répertc
   int            common_flag=0         // définit le domaine de recherches
   );
```

Paramètres

file_filter

[in] Le filtre de la recherche. On peut indiquer le sous-répertoire dans le filtre (ou la séquence des sous-répertoires inclus) concernant le répertoire \Files, où il est nécessaire de passer l'excédent des fichiers.

returned_filename

[out] Le paramètre rendu, où en cas de la réussite se place le nom du premier fichier trouvé ou du sous-répertoire. Only the file name is returned (including the extension), the directories and subdirectories are not included no matter if they are specified or not in the search filter.

common_flag

[in] Le drapeau, déterminant l'endroit du fichier. Si common_flag=FILE_COMMON, le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le fichier se trouve dans le dossier local.

La valeur rendue

Rend le handle de l'objet de la recherche, lequel il est nécessaire d'utiliser pour un balayage ultérieur des fichiers et des sous-répertoires par la fonction FileFindNext(), ou INVALID_HANDLE dans le cas où il n'y a pas d'aucun fichier et le sous-répertoire, correspondant au filtre (dans le cas particulier - le répertoire est vide). Après la fin de la recherche il est nécessaire de fermer le handle à l'aide de la fonction FileFindClose().

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//--- display the window of input parameters when launching the script
#property script_show_inputs
//--- filter
input string InpFilter="Dir1\\*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   string int_dir="";
   int    i=1,pos=0,last_pos=-1;
//--- search for the last backslash
   while(!IsStopped())
     {
      pos=StringFind(InpFilter,"\\",pos+1);
      if(pos>=0)
         last_pos=pos;
      else
         break;
     }
//--- the filter contains the folder name
   if(last_pos>=0)
      int_dir=StringSubstr(InpFilter,0,last_pos+1);
//--- get the search handle in the root of the local folder
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- check if the FileFindFirst() is executed successfully
   if(search_handle!=INVALID_HANDLE)
     {
      //--- in a cycle, check if the passed strings are the names of files or directo
      do
        {
         ResetLastError();
         //--- if it's a file, the function returns true, and if it's a directory, it
         FileIsExist(int_dir+file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Di
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- close the search handle
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

Voir aussi

FileFindNext, FileFindClose

© 2000-2015, MetaQuotes Software Corp.

FileFindNext

Continue la recherche commencée par la fonction FileFindFirst().

```
bool  FileFindNext(
   long      search_handle,         // handle de la recherche
   string&   returned_filename      // le nom du fichier trouvé ou du sous-répertoire
   );
```

Paramètres

search_handle

[in] Le handle de la recherche, reçu par la fonction FileFindFirst().

returned_filename

[out] Le nom du fichier suivant trouvé ou du répertoire. Only the file name is returned (including the extension), the directories and subdirectories are not included no matter if they are specified or not in the search filter.

La valeur rendue

Rend true en cas du succès, autrement rend false.

Exemple:

```
//--- affichons la fenêtre des paramètres d'entrée pendant le lancement du script
#property script_show_inputs
//--- le filtre
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- la réception du handle de la recherche dans la racine du dossier local
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- vérifions si la fonction FileFindFirst() a travaillé avec succès
   if(search_handle!=INVALID_HANDLE)
     {
      //--- dans le cycle vérifions si les chaînes transmises sont les noms des fichie
      do
        {
         ResetLastError();
         //--- si c'est le fichier, la fonction rendra true, et si c'est le répertoire
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Di
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- fermons le handle de la recherche
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

© 2000-2015, MetaQuotes Software Corp.

Voir aussi

FileFindFirst, FileFindClose

© 2000-2015, MetaQuotes Software Corp.

FileFindClose

Ferme le handle de la recherche.

```
void  FileFindClose(
   long  search_handle      // handle de la recherche
   );
```

Paramètres

search_handle

[in] Le handle de la recherche, reçu par la fonction FileFindFirst().

La valeur rendue

Il n'y a pas de valeur rendue.

Note

Il est nécessaire d'appeler la fonction pour désallouer les ressources systémiques.

Exemple:

```
//--- affichons la fenêtre des paramètres d'entrée pendant le lancement du script
#property script_show_inputs
//--- le filtre
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- la réception du handle de la recherche dans la racine du dossier local
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- vérifions si la fonction FileFindFirst() a travaillé avec succès
   if(search_handle!=INVALID_HANDLE)
     {
      //--- dans le cycle vérifions si les chaînes transmises sont les noms des fichie
      do
        {
         ResetLastError();
         //--- si c'est le fichier, la fonction rendra true, et si c'est le répertoire
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==5018 ? "Directory" : "File"
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- fermons le handle de la recherche
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

Voir aussi

FileFindFirst, FileFindNext

© 2000-2015, MetaQuotes Software Corp.

FileIsExist

Vérifie l'existence d'un fichier.

```
bool  FileIsExist(
   const string  file_name,        // nom du fichier
   int           common_flag=0     // domaine de recherche
   );
```

Paramètres

file_name

[in] Le nom du fichier vérifié.

common_flag=0

[in] Le drapeau, déterminant l'endroit du fichier. Si common_flag=FILE_COMMON, le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le fichier se trouve dans le dossier local.

La valeur rendue

Rend true, si le fichier indiqué existe.

Note

Le fichier vérifié peut être le sous-répertoire. Dans ce cas la fonction FileIsExist() rendra false, et l'erreur 5018 - "Ce n'est pas un fichier, c'est un répertoire" (regardez l'exemple pour la fonction FileFindFirst) sera inscrite dans une variable _LastError.

On contrôle strictement le travail avec les fichiers dans la langue MQL5 pour des raisons de sécurité. Les fichiers, avec lesquels on mène des opérations de fichier par les moyens de la langue MQL5, ne peuvent pas se trouver en dehors du sandbox de fichier".

Si common_flag=FILE_COMMON, la fonction cherche le fichier indiqué dans le dossier commun de tous les terminaux de client \Terminal\Common\Files, dans le cas contraire la fonction cherche le fichier dans le dossier local (MQL5\Files ou MQL5\Tester\Files en cas du test).

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//---la date pour les anciens fichiers
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // la variable pour stocker les noms des fichiers
   string   filter="*.txt"; // le filtre pour rechercher les fichiers
   datetime create_date;    // la date de la création du fichier
   string   files[];        // la liste des noms des fichiers
   int      def_size=25;    // la taille du tableau par défaut
   int      size=0;         // le nombre de fichiers
//--- allouons la mémoire pour un tableau
   ArrayResize(files,def_size);
//--- la réception du handle de la recherche dans la racine du dossier local
   long search_handle=FileFindFirst(filter,file_name);
//--- vérifions, si la fonction FileFindFirst() a travaillé avec succès
   if(search_handle!=INVALID_HANDLE)
     {
      //--- trions les fichiers dans le cycle
      do
        {
         files[size]=file_name;
         //--- augmentons la taille du tableau
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- oblitérons la valeur de l'erreur
         ResetLastError();
         //---recevrons la date de la création du fichier
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- vérifions si le fichier est ancien
         if(create_date<InpFilesDate)
           {
            PrintFormat("Le fichier %s est supprimé!",file_name);
            //--- supprimons un ancien fichier
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- fermons le handle de la recherche
      FileFindClose(search_handle);
     }
   else
     {
      Print("Files not found!");
      return;
     }
//--- vérifions quels fichiers sont restés
   PrintFormat("Les résultats:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("Le fichier %s existe!",files[i]);
      else
```

© 2000-2015, MetaQuotes Software Corp.

```
            PrintFormat("Le fichier %s est supprimé!",files[i]);
        }
    }
```

Voir aussi

FileFindFirst

© 2000-2015, MetaQuotes Software Corp.

FileOpen

Ouvre un fichier avec un nom et un drapeau indiqués.

```
int  FileOpen(
   string  file_name,           // nom du fichier
   int     open_flags,          // combinaison des drapeaux
   short   delimiter='\t',      // délimiteur
   uint    codepage=CP_ACP      // page de code
   );
```

Paramètres

file_name

[in] Le nom du fichier ouvert peut contenir les sous-dossiers. Si le fichier s'ouvre pour l'enregistrement, les sous-dossiers indiqués seront créés en cas de leur absence.

open_flags

[in] la combinaison des drapeaux, définissant le mode du travail avec le fichier. Les drapeaux sont définis à la manière suivante:
FILE_READ le fichier s'ouvre pour la lecture
FILE_WRITE le fichier s'ouvre pour l'enregistrement
FILE_BIN le mode binaire de la lecture- l'enregistrement (sans conversion de la chaîne et à la chaîne)
FILE_CSV le fichier du type csv (tous les éléments inscrits sont transformés vers les chaînes du type correspondant, unicode ou ansi, et se divisent par le délimiteur)
FILE_TXT Un simple fichier du texte (le même csv, mais le délimiteur n'est pas pris en considération)
FILE_ANSI les chaînes du type ANSI (les caractères d'un byte)
FILE_UNICODE les chaînes du type UNICODE (les caractères de deux bytes)
FILE_SHARE_READ l'accès commun selon la lecture du côté de quelques programmes
FILE_SHARE_WRITE l'accès commun selon l'enregistrement du côté de quelques programmes
FILE_COMMON la disposition du fichier dans le dossier commun de tous les terminaux de client \Terminal\Common\Files

delimiter='\t'

[in] la valeur utilisée à titre du délimiteur dans les fichiers txt ou csv. Si pour le fichier csv le délimiteur n'est pas indiqué, on utilise le caractère de la tabulation par défaut. Si pour le fichier txt le délimiteur n'est pas indiqué, aucun délimiteur n'est pas utilisé. Si à titre du délimiteur on spécifie évidemment la valeur 0, aucun délimiteur n'est pas utilisé.

codepage=CP_ACP

[in] La valeur de la page de code. Pour des pages de code les plus utilisées on prévoit les constantes correspondantes.

La valeur rendue

En cas de l'ouverture réussie la fonction rend le handle du fichier, qui est utilisé pour l'accès aux données du fichier. En cas de l'échec rend INVALID_HANDLE.

Note

On contrôle strictement le travail avec les fichiers dans la langue MQL5 pour des raisons de sécurité.

© 2000-2015, MetaQuotes Software Corp.

Les fichiers, avec lesquels on mène des opérations de fichier par les moyens de la langue MQL5, ne peuvent pas se trouver en dehors du sandbox de fichier".

Le fichier s'ouvre dans le dossier du terminal de client dans le sous-dossier MQL5\files (ou le catalogue_de l'agent_du test\MQL5\files en cas du test). Si parmi les drapeaux est indiqué FILE_COMMON, le fichier s'ouvre dans le dossier commun de tous les terminaux de client MetaTrader5.

On peut ouvrir "les canaux nommés" selon les règles suivantes:

- Le nom du canal - c'est une chaîne qui doit avoir l'air: "\\servername\pipe\pipename", où servername - c'est le nom du serveur au réseau, et pipename - c'est le nom du canal. Si les canaux sont utilisés sur le même ordinateur, le nom du serveur peut être omis, mais au lieu de lui il faut mettre le point: "\\.\pipe\pipename". Un client qui tente de se connecter au canal, doit savoir son nom.
- Il faur appeler FileFlush() et FileSeek() au début du fichier entre les opérations successives de la lecture du canal et de l'inscription au canal.

Dans ces chaînes on utilise le symbole spécial la ligne inverse oblique ' \', c'est pourquoi à l'écriture du nom au programme MQL4 ' \' il est nécessaire de doubler, c'est-à-dire l'exemple cité ci-dessus il faut écrire dans le code comme" "\\\\servername\\pipe\\pipename".

On peut trouver plus de détails sur le travail avec les canaux nommés dans l'article "Communicating With MetaTrader 5 Using Named Pipes Without Using DLLs".

Exemple:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- moyen incorrect de l'ouverture du fichier
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   string filename=terminal_data_path+"\\MQL5\\Files\\"+"fractals.csv";
   int filehandle=FileOpen(filename,FILE_WRITE|FILE_CSV);
   if(filehandle<0)
     {
      Print("tentative ratйe d'ouvrir un fichier par un chemin absolu");
      Print("Code de l'erreur",GetLastError());
     }
//--- la faзon correcte de travailler dans "le bac a sable de fichier"
   ResetLastError();
   filehandle=FileOpen("fractals.csv",FILE_WRITE|FILE_CSV);
   if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
      Print("FileOpen OK");
     }
   else Print("Opйration FileOpen est ratйe, l'erreur ",GetLastError());
//--- encore un exemple avec la crйation du rйpertoire inclus dans MQL5\Files\
```

© 2000-2015, MetaQuotes Software Corp.

```
   string subfolder="Research";
   filehandle=FileOpen(subfolder+"\\fractals.txt",FILE_WRITE|FILE_CSV);
      if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
      Print("Fichier doit кtre cгйй dans le dossier "+terminal_data_path+"\\"+subfolde
     }
   else Print("File open failed, error ",GetLastError());
  }
```

Voir aussi

Utilisation de la page de code, FileFindFirst, FolderCreate, Les drapeaux de l'ouverture des fichiers

© 2000-2015, MetaQuotes Software Corp.

FileClose

Ferme le fichier auparavant ouvert par la fonction FileOpen().

```
void  FileClose(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendue par la fonction FileOpen().

La valeur rendue

Il n'y a pas de valeur rendue.

Exemple:

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="file.txt";     // le nom du fichier
input string InpDirectoryName="Data";    // le nom du répertoire
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imprimons le chemin vers le dossier dans lequel on va travailler
   PrintFormat("Travaillons dans le dossier %s\\Files\\",TerminalInfoString(TERMINAL_I
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrirons le fichier pour la lecture (si le fichier n'existe pas, une erreur se
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      //---imprimons le contenu du fichier
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
      PrintFormat("L'erreur, le code = %d",GetLastError());
  }
```

© 2000-2015, MetaQuotes Software Corp.

FileCopy

Copie le fichier original d'un répertoire local ou commun à un autre fichier.

```
bool  FileCopy(
   const string  src_file_name,     // nom du fichier-source
   int           common_flag,       // lieu de l'action
   const string  dst_file_name,     // nom du fichier de la destination
   int           mode_flags         // mode de l'accès
   );
```

Paramètres

src_file_name

[in] Le nom du fichier pour le copiage.

common_flag

[in] Le drapeau, déterminant l'endroit du fichier. Si common_flag=FILE_COMMON, le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le fichier se trouve dans le dossier local (par exemple, *common_flag=0*).

dst_file_name

[in] Le nom de fichier de résultat.

mode_flags

[in] Les drapeaux de l'accès. Le paramètre peut contenir seulement 2 drapeaux: FILE_REWRITE et/ou FILE_COMMON - les autres drapeaux sont ignorés. Si le fichier existe déjà et le drapeau FILE_REWRITE n'a pas été spécifié, donc le fichier ne sera pas réécrit et la fonction rendra false.

La valeur rendue

En cas de l'échec la fonction rend false.

Note

Si un nouveau fichier existait, le copiage sera produit en fonction de la présence du drapeau FILE_REWRITE dans la valeur du paramètre mode_flags.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//---affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpSrc="source.txt";       // la source
input string InpDst="destination.txt";  // la copie
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- affichons le contenu de la source (il doit exister)
   if(!FileDisplay(InpSrc))
      return;
//--- vérifions, s'il y a déjà un fichier de la copie (ne doit pas être créé)
   if(!FileDisplay(InpDst))
     {
      //--- le fichier de la copie n'existe pas, le copiage sans drapeau FILE_REWRITE
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("File is copied!");
      else
         Print("File is not copied!");
     }
   else
     {
      //--- le fichier de la copie existe déjà, essaierons copier sans drapeau FILE_RE
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("File is copied!");
      else
         Print("File is not copied!");
      //---le contenu du fichier InpDst sera le même
      FileDisplay(InpDst);
      //--- copions encore une fois avec le drapeau FILE_REWRITE (le copiage juste à l
      if(FileCopy(InpSrc,0,InpDst,FILE_REWRITE))
         Print("File is copied!");
      else
         Print("File is not copied!");
     }
//--- ont reçu la copie du fichier InpSrc
   FileDisplay(InpDst);
  }
//+------------------------------------------------------------------+
//| la lecture du contenu du fichier                                 |
//+------------------------------------------------------------------+
bool FileDisplay(const string file_name)
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrons le fichier
   int file_handle=FileOpen(file_name,FILE_READ|FILE_TXT|InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- affichons le contenu d'un fichier dans une boucle
      Print("+----------------------+");
      PrintFormat("File name = %s",file_name);
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      Print("+----------------------+");
      //--- fermons le fichier
      FileClose(file_handle);
      return(true);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
//--- on n'a pas réussi à ouvrir le fichier
   PrintFormat("%s is not opened, error = %d",file_name,GetLastError());
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

FileDelete

Supprime le fichier indiqué dans le dossier local du terminal de client.

```
bool  FileDelete(
   const string  file_name,      // nom du fichier supprimé
   int           common_flag=0   // endroit du fichier supprimé
   );
```

Paramètres

file_name

[in] Le nom du fichier.

common_flag=0

[in] Le drapeau, déterminant l'endroit du fichier. Si common_flag=FILE_COMMON, le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le fichier se trouve dans le dossier local.

La valeur rendue

En cas de l'échec la fonction rend false.

Note

On contrôle strictement le travail avec les fichiers dans la langue MQL5 pour des raisons de sécurité. Les fichiers, avec lesquels on mène des opérations de fichier par les moyens de la langue MQL5, ne peuvent pas se trouver en dehors du sandbox de fichier".

Supprime le fichier indiqué dans le dossier local du terminal de client (MQL5\Files ou MQL5\Tester \Files en cas du test). Si common_flag=FILE_COMMON est indiqué, la fonction supprime le fichier du dossier total de tous les terminaux de client.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//---la date pour les anciens fichiers
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // la variable pour stocker les noms des fichiers
   string   filter="*.txt"; // le filtre pour rechercher les fichiers
   datetime create_date;    // la date de la création du fichier
   string   files[];        // la liste des noms des fichiers
   int      def_size=25;    // la taille du tableau par défaut
   int      size=0;         // le nombre de fichiers
//--- allouons la mémoire pour un tableau
   ArrayResize(files,def_size);
//--- la réception du handle de la recherche dans la racine du dossier local
   long search_handle=FileFindFirst(filter,file_name);
//--- vérifions, si la fonction FileFindFirst() a travaillé avec succès
   if(search_handle!=INVALID_HANDLE)
     {
      //--- trions les fichiers dans le cycle
      do
        {
         files[size]=file_name;
         //--- augmentons la taille du tableau
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- oblitérons la valeur de l'erreur
         ResetLastError();
         //---recevrons la date de la création du fichier
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- vérifions si le fichier est ancien
         if(create_date<InpFilesDate)
           {
            PrintFormat("Le fichier %s est supprimé!",file_name);
            //--- supprimons un ancien fichier
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- fermons le handle de la recherche
      FileFindClose(search_handle);
     }
   else
     {
      Print("Files not found!");
      return;
     }
//--- vérifions quels fichiers sont restés
   PrintFormat("Les résultats:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("Le fichier %s existe!",files[i]);
      else
```

© 2000-2015, MetaQuotes Software Corp.

```
            PrintFormat("Le fichier %s est supprimé!",files[i]);
        }
    }
```

© 2000-2015, MetaQuotes Software Corp.

FileMove

Déplace le fichier du dossier local ou total à un autre dossier.

```
bool  FileMove(
   const string  src_file_name,     // nom du fichier pour l'opération du déplacement
   int           common_flag,       // lieu de l'action
   const string  dst_file_name,     // nom du fichier de la destination
   int           mode_flags         // mode de l'accès
   );
```

Paramètres

src_file_name

[in] Le nom du fichier pour le déplacement/la renomination.

common_flag

[in] Le drapeau, déterminant l'endroit du fichier. Si common_flag=FILE_COMMON, le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files. Dans le cas contraire le fichier se trouve dans le dossier local (*common_flag=0*).

dst_file_name

[in] Le nom de fichier de résultat.

mode_flags

[in] Les drapeaux de l'accès. Le paramètre peut contenir seulement 2 drapeaux: FILE_REWRITE et/ou FILE_COMMON - les autres drapeaux sont ignorés. Si le fichier existe déjà et le drapeau FILE_REWRITE, n'a pas été spécifié, donc le fichier ne sera pas réécrit et la fonction rendra false.

La valeur rendue

En cas de l'échec la fonction rend false.

Note

On contrôle strictement le travail avec les fichiers dans la langue MQL5 pour des raisons de sécurité. Les fichiers, avec lesquels on mène des opérations de fichier par les moyens de la langue MQL5, ne peuvent pas se trouver en dehors du sandbox de fichier".

Si un nouveau fichier existait, le copiage sera produit en fonction de la présence du drapeau FILE_REWRITE dans la valeur du paramètre mode_flags.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpSrcName="data.txt";
input string InpDstName="newdata.txt";
input string InpSrcDirectory="SomeFolder";
input string InpDstDirectory="OtherFolder";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string local=TerminalInfoString(TERMINAL_DATA_PATH);
   string common=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- recevrons les voies vers les fichiers
   string src_path;
   string dst_path;
   StringConcatenate(src_path,InpSrcDirectory,"//",InpSrcName);
   StringConcatenate(dst_path,InpDstDirectory,"//",InpDstName);
//--- vérifions s'il y a un fichier source (si elle n'existe pas - la sortie)
   if(FileIsExist(src_path))
      PrintFormat("%s file exists in the %s\\Files\\%s folder",InpSrcName,local,InpSrc
   else
     {
      PrintFormat("Error, %s source file not found",InpSrcName);
      return;
     }
//--- vérifions, s'il y a déjà un fichier du résultat
   if(FileIsExist(dst_path,FILE_COMMON))
     {
      PrintFormat("%s file exists in the %s\\Files\\%s folder",InpDstName,common,InpDs
      //--- le fichier existe, il faut passer le déplacement avec le drapeau FILE_REWR
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON|FILE_REWRITE))
         PrintFormat("%s file moved",InpSrcName);
      else
         PrintFormat("Error! Code = %d",GetLastError());
     }
   else
     {
      PrintFormat("%s file does not exist in the %s\\Files\\%s folder",InpDstName,comm
      //--- le fichier n'existe pas, il faut passer le déplacement sans drapeau FILE_R
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON))
         PrintFormat("%s file moved",InpSrcName);
      else
         PrintFormat("Error! Code = %d",GetLastError());
     }
//--- maintenant le fichier est déplacé, vérifions ça
   if(FileIsExist(dst_path,FILE_COMMON) && !FileIsExist(src_path,0))
      Print("Success!");
   else
      Print("Error!");
  }
```

Voir aussi

FileIsExist

© 2000-2015, MetaQuotes Software Corp.

FileFlush

Écrit à un disque toutes les données restant dans le tampon du fichier d'entrée-de sortie.

```
void  FileFlush(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

Il n'y a pas de valeur rendue.

Note

A l'exécution de l'opération de l'inscription au fichier, les données peuvent s'y trouver physiquement seulement dans un certain temps. Pour que les données se sauvegardent tout de suite dans un fichier, il faut utiliser la fonction FileFlush(). Si ne pas utiliser la fonction, la partie des données qui ne se sont pas encore sur le disque s'inscrit forcément là-bas seulement à la clôture du fichier par la fonction FileClose().

Il est nécessaire d'utiliser la fonction, quand les données inscrites ont la valeur définie. Il faut prendre en considération que l'appel fréquent de la fonction peut affecter la vitesse du travail du programme.

Il est nécessaire d'appeler la fonction FileFlush() entre les opérations de la lecture du fichier et l'enregistrement au fichier.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//---le nom du fichier pour l'inscription
input string InpFileName="example.csv"; // le nom du fichier
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrons le fichier
   int file_handle=FileOpen(InpFileName,FILE_READ|FILE_WRITE|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- inscrirons les données au fichier
      for(int i=0;i<1000;i++)
        {
         //--- appelons la fonction de l'inscription
         FileWrite(file_handle,TimeCurrent(),SymbolInfoDouble(Symbol(),SYMBOL_BID),Sym
         //--- enregistrons les données sur le disque sur chaque 128-ième itération
         if((i & 127)==127)
           {
            //--- maintenant les données se trouvent dans le fichier, et à l'erreur ci
            FileFlush(file_handle);
            PrintFormat("i = %d, OK",i);
           }
         //--- le retard à 0.01 seconde
         Sleep(10);
        }
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
      PrintFormat("L'erreur, le code = %d",GetLastError());
  }
```

Voir aussi

FileClose

© 2000-2015, MetaQuotes Software Corp.

FileGetInteger

Reçoit une propriété entière du fichier. Il y a 2 variantes de la fonction.

1. La réception des propriétés selon le handle du fichier.

```
long  FileGetInteger(
   int                          file_handle,    // le handle du fichier
   ENUM_FILE_PROPERTY_INTEGER   property_id     // l'identificateur de la propriété
   );
```

2. La réception des propriétés selon le nom du fichier.

```
long  FileGetInteger(
   const string                 file_name,            // le nom du fichier
   ENUM_FILE_PROPERTY_INTEGER   property_id,          // l'identificateur de la propri
   bool                         common_folder=false   // le fichier est affiché dans u
   );                                                 // ou dans le dossier commun de t
```

Paramètres

file_handle

[in] Le descripteur de fichier, rendu par la fonction FileOpen().

file_name

[in] Le nom du fichier.

property_id

[in] L'identificateur de la propriété du fichier. La valeur peut être l'une des valeurs de l'énumérationENUM_FILE_PROPERTY_INTEGER. Si on utilise une deuxième variante de la fonction, vous pouvez recevoir la valeur seulement des propriétés suivantes: FILE_EXISTS, FILE_CREATE_DATE, FILE_MODIFY_DATE, FILE_ACCESS_DATE et FILE_SIZE.

common_folder=false

[in] Indique à l'emplacement du fichier. Si le paramètre est égal false, on examine le répertoire des données du terminal, sinon il est supposé que le fichier se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common\Files (FILE_COMMON).

La valeur rendue

La valeur de la propriété. En cas de l'erreur rend -1, pour recevoir un code de l'erreur il faut appeler la fonction GetLastError().

Si on indique le répertoire à la réception des propriétés selon le nom, la fonction mettra l'erreur 5018 (ERR_MQL_FILE_IS_DIRECTORY) dans tous les cas, en cela la valeur rendue sera correcte.

Note

La fonction change toujours le code de l'erreur. En cas de succès, le code de l'erreur est remis à zéro.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="data.csv";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string path=InpDirectoryName+"//"+InpFileName;
   long   l=0;
//--- ouvrons le fichier
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_CSV);
   if(handle!=INVALID_HANDLE)
     {
      //--- imprimons toute l'information sur le fichier
      Print(InpFileName," file info:");
      FileInfo(handle,FILE_EXISTS,l,"bool");
      FileInfo(handle,FILE_CREATE_DATE,l,"date");
      FileInfo(handle,FILE_MODIFY_DATE,l,"date");
      FileInfo(handle,FILE_ACCESS_DATE,l,"date");
      FileInfo(handle,FILE_SIZE,l,"other");
      FileInfo(handle,FILE_POSITION,l,"other");
      FileInfo(handle,FILE_END,l,"bool");
      FileInfo(handle,FILE_IS_COMMON,l,"bool");
      FileInfo(handle,FILE_IS_TEXT,l,"bool");
      FileInfo(handle,FILE_IS_BINARY,l,"bool");
      FileInfo(handle,FILE_IS_CSV,l,"bool");
      FileInfo(handle,FILE_IS_ANSI,l,"bool");
      FileInfo(handle,FILE_IS_READABLE,l,"bool");
      FileInfo(handle,FILE_IS_WRITABLE,l,"bool");
      //--- fermons le fichier
      FileClose(handle);
     }
   else
      PrintFormat("%s file is not opened, ErrorCode = %d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------+
//| L'affichage de la valeur des propriétés du fichier               |
//+------------------------------------------------------------------+
void FileInfo(const int handle,const ENUM_FILE_PROPERTY_INTEGER id,
              long l,const string type)
  {
//--- recevrons la valeur de la propriété
   ResetLastError();
   if((l=FileGetInteger(handle,id))!=-1)
     {
      //--- La valeur est reçue, l'affichons au format juste
      if(!StringCompare(type,"bool"))
         Print(EnumToString(id)," = ",l ? "true" : "false");
      if(!StringCompare(type,"date"))
         Print(EnumToString(id)," = ",(datetime)l);
      if(!StringCompare(type,"other"))
         Print(EnumToString(id)," = ",l);
     }
   else
      Print("Error, Code = ",GetLastError());
  }
```

© 2000-2015, MetaQuotes Software Corp.

Voir aussi

Les opérations de fichier, Les propriétés des fichiers

© 2000-2015, MetaQuotes Software Corp.

FileIsEnding

Définit la fin du fichier en train de la lecture.

```
bool  FileIsEnding(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La fonction rend true dans le cas, si en train de la lecture ou du déplacement de l'indication de fichier atteindront la fin du fichier.

Note

Pour la définition de la fin du fichier, la fonction essaie de lire la chaîne suivante du fichier. Si elle n'existe pas, la fonction rend true, autrement rend false.

Exemple:

```
//--- affichons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="file.txt";     // le nom du fichier
input string InpDirectoryName="Data";    // le nom du répertoire
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imprimons le chemin vers le dossier dans lequel on va travailler
   PrintFormat("Travaillons dans le dossier %s\\Files\\",TerminalInfoString(TERMINAL_I
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrirons le fichier pour la lecture (si le fichier n'existe pas, une erreur ser
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      //---imprimons le contenu du fichier
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
      PrintFormat("L'erreur, le code = %d",GetLastError());
  }
```

© 2000-2015, MetaQuotes Software Corp.

FileIsLineEnding

Définit la fin de la ligne dans le fichier du texte en train de la lecture.

```
bool  FileIsLineEnding(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La fonction rend true dans le cas, si en train de la lecture des fichiers txt ou csv atteint la fin de la ligne (les caractères CR-LF).

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteString)

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                        Demo_FileIsLineEnding.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- les paramètres pour la lecture des données
input string InpFileName="RSI.csv";   // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les tampons d'indicateur
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- les variables du surachat
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- les variables de la survente
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- les variables des tailles des tableaux par défaut
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- allouons la mémoire pour les tableaux
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier : %s\\Files\\",TerminalInfoString(TERMINAL_
      double value;
      //--- lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //---lisons la première signification dans la chaîne
         value=FileReadNumber(file_handle);
         //--- lisons aux tableaux différents en fonction du résultat de la fonction
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
```

© 2000-2015, MetaQuotes Software Corp.

```
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- le rattachement des tableaux
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
//---- l'établissement des valeurs de l'indicateur, qui ne seront pas vus sur le graph
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| La lecture des données de la chaîne du fichier
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- lisons jusqu'à ce que nous arrivons à la fin d'une chaîne ou d'un fichier
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- rapprocherons le chariot, ayant lu le nombre
      if(flag)
         FileReadNumber(file_handle);
      //--- retenons la date courante
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- augmentons en cas de nécessité la taille du tableau
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- on a passé la première itération
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2015, MetaQuotes Software Corp.

```
{
```

© 2000-2015, MetaQuotes Software Corp.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- si les dates correspondent, la barre se trouve dans la zone de surac
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - la couleur rouge
               color_buff[i]=0;
               //--- augmentons le compteur
               ovb_ind=j+1;
               break;
              }
           }
      //--- la vérification s'il y a encore des données
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- si les dates correspondent, la barre se trouve dans la zone de la su
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - la couleur bleue
               color_buff[i]=1;
               //---augmentons le compteur
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Le gestionnaire de l'événement ChartEvent                        |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
```

© 2000-2015, MetaQuotes Software Corp.

```
                     )
  {
//---changeons l'épaisseur de l'indicateur en fonction de l'échelle
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Voir aussi

FileWriteString

© 2000-2015, MetaQuotes Software Corp.

FileReadArray

Lit les tableaux de tous les types, sauf les tableaux de chaîne (peut être un tableau des structures qui ne contient pas les chaînes et les tableaux dynamiques), du fichier binaire de la position courante de l'indicateur de fichier.

```
uint  FileReadArray(
   int     file_handle,              // handle du fichier
   void&   array[],                  // tableau pour l'enregistrement
   int     start=0,                  // de quelle position du tableau écrire
   int     count=WHOLE_ARRAY         // combien de lire
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

array[]

[out] Le tableau où les données seront chargées.

start=0

[in] La position de départ pour l'enregistrement au tableau.

count=WHOLE_ARRAY

[in] Le nombre d'éléments pour la lecture.

La valeur rendue

Le nombre d'éléments lus. Par défaut, lit tout le tableau (count=WHOLE_ARRAY).

Note

Le tableau de chaîne peut être lu seulement du fichier du type TXT. En cas de nécessité la fonction tente d'augmenter la grandeur du tableau.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteArray)

© 2000-2015, MetaQuotes Software Corp.

```
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| La structure pour la sauvegarde des données sur les prix
//+------------------------------------------------------------------+
struct prices
  {
   datetime          date; // la date
   double            bid;  // le prix bid
   double            ask;  // le prix ask
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- un tableau de structure
   prices arr[];
//--- la voie vers le fichier
   string path=InpDirectoryName+"//"+InpFileName;
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(path,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- lisons toutes les données du fichier au tableau
      FileReadArray(file_handle,arr);
      //--- recevrons la taille du tableau
      int size=ArraySize(arr);
      //--- imprimons les données du tableau
      for(int i=0;i<size;i++)
         Print("Date = ",arr[i].date," Bid = ",arr[i].bid," Ask = ",arr[i].ask);
      Print("Total data = ",size);
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
      Print("File open failed, error ",GetLastError());
  }
```

Voir aussi

Les variables, FileWriteArray

© 2000-2015, MetaQuotes Software Corp.

FileReadBool

Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type bool.

```
bool  FileReadBool(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La ligne lue peut avoir les valeurs "true", "false" ou la représentation symbolique des nombres entières "0" ou "1". La valeur non zéro se transforme vers logique true. La fonction rend la valeur reçue transformée.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWrite)

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                              Demo_FileReadBool.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- les paramètres pour la lecture des données
input string InpFileName="MACD.csv";  // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les variables globales
int      ind=0;       // l'index
double   upbuff[];    // le tampon d'indicateur des flèches en haut
double   downbuff[];  // le tampon d'indicateur des flèches en bas
bool     sign_buff[]; // le tableau des signaux (true - l'achat, false - la vente)
datetime time_buff[]; // le tableau du temps de l'arrivée des signaux
int      size=0;      // la taille des tableau des signaux
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      //--- lisons d'abord le nombre de signaux
      size=(int)FileReadNumber(file_handle);
      //--- allouons la mémoire pour les tableaux
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- lisons des données du fichier
      for(int i=0;i<size;i++)
        {
         //--- le temps du signal
         time_buff[i]=FileReadDatetime(file_handle);
         //--- la valeur du signal
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
       return(INIT_FAILED);
      }
//--- le rattachement des tableaux
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- spécifions le code du symbole pour la rendu dans PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- l'établissement des valeurs de l'indicateur qui ne seront pas visibles sur le g
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               //--- dessinons la flèche en fonction du signal
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Le type bool, FileWrite

© 2000-2015, MetaQuotes Software Corp.

FileReadDatetime

Lit du fichier du type CSV la chaîne d'un des formats: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" ou "HH:MI:SS" - et la transforme en valeur du type datetime.

```
datetime  FileReadDatetime(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type datetime.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWrite)

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                        Demo_FileReadDateTime.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- les paramètres pour la lecture des données
input string InpFileName="MACD.csv";  // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les variables globales
int      ind=0;       // l'index
double   upbuff[];    // le tampon d'indicateur des flèches en haut
double   downbuff[];  // le tampon d'indicateur des flèches en bas
bool     sign_buff[]; // le tableau des signaux (true - l'achat, false - la vente)
datetime time_buff[]; // le tableau du temps de l'arrivée des signaux
int      size=0;      // la taille des tableau des signaux
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      //--- lisons d'abord le nombre de signaux
      size=(int)FileReadNumber(file_handle);
      //--- allouons la mémoire pour les tableaux
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- lisons des données du fichier
      for(int i=0;i<size;i++)
        {
         //--- le temps du signal
         time_buff[i]=FileReadDatetime(file_handle);
         //--- la valeur du signal
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- fermons le fichier
      FileClose(file_handle);
     }
   else
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
       return(INIT_FAILED);
      }
//--- le rattachement des tableaux
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- spécifions le code du symbole pour la rendu dans PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- l'établissement des valeurs de l'indicateur qui ne seront pas visibles sur le g
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               //--- dessinons la flèche en fonction du signal
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Le type datetime, StringToTime, TimeToString, FileWrite

© 2000-2015, MetaQuotes Software Corp.

FileReadDouble

Lit le nombre de l'exactitude double avec une virgule flottante (double) du fichier binaire de la position courante de l'indicateur de fichier.

```
double  FileReadDouble(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type double.

Note

Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteDouble)

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- les paramètres pour la lecture des données
input string InpFileName="MA.csv";     // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les variables globales
int      ind=0;
int      size=0;
double   ma_buff[];
datetime time_buff[];
//--- indicator buffer
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- lisons d'abord combien de données sont dans le fichier
      size=(int)FileReadDouble(file_handle);
      //--- allouons la mémoire pour les tableaux
      ArrayResize(ma_buff,size);
      ArrayResize(time_buff,size);
      //--- lisons les données du fichier
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadDouble(file_handle);
         ma_buff[i]=FileReadDouble(file_handle);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- Le rattachement du tableau au tampon d'indicateur avec l'indice 0
```

© 2000-2015, MetaQuotes Software Corp.

```
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- l'établissement des valeurs de l'indicateur qui ne seront pas visibles sur le g
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      buff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               buff[i]=ma_buff[j];
               //---augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Les types réels (double, float), StringToDouble, DoubleToString, FileWriteDouble

© 2000-2015, MetaQuotes Software Corp.

FileReadFloat

Lit le nombre de l'exactitude simple avec la virgule flottante (float) de la position courante du fichier binaire.

```
float  FileReadFloat(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type float.

Note

Pour recevoir l'information sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteFloat)

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "CloseLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- les paramètres pour la lecture des données
input string InpFileName="Close.bin"; // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les variables globales
int      ind=0;
int      size=0;
double   close_buff[];
datetime time_buff[];
//--- indicator buffers
double   buff[];
double   color_buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- allouons la mémoire pour les tableaux
   ArrayResize(close_buff,def_size);
   ArrayResize(time_buff,def_size);
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat(Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //--- lisons la signification du temps et du prix
         time_buff[size]=(datetime)FileReadDouble(file_handle);
         close_buff[size]=(double)FileReadFloat(file_handle);
         size++;
         //--- augmenons les tailles des tableaux s'ils sont remplis
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(close_buff,def_size);
            ArrayResize(time_buff,def_size);
           }
        }
      //--- fermons le fichier
      FileClose(file_handle);
```

© 2000-2015, MetaQuotes Software Corp.

```
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,buff,INDICATOR_DATA);
   SetIndexBuffer(1,color_buff,INDICATOR_COLOR_INDEX);
//---- l'établissement des valeurs de l'indicateur, qui ne seront pas vus sur le graph
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      buff[i]=0;
      color_buff[i]=0; // la couleur rouge par défaut
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               //--- recevrons le prix
               buff[i]=close_buff[j];
               //--- si le prix actuel est plus que le précédente, la couleur est bleu
               if(buff[i-1]>buff[i])
                  color_buff[i]=1;
               //--- augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

© 2000-2015, MetaQuotes Software Corp.

Les types réels (double, float), FileReadDouble, FileWriteFloat

© 2000-2015, MetaQuotes Software Corp.

FileReadInteger

Lit du fichier binaire la valeur du int, short ou char en fonction de la longueur indiquée dans les bytes. La lecture est produite de la position courante de l'indicateur de fichier.

```
int  FileReadInteger(
   int  file_handle,         // handle du fichier
   int  size=INT_VALUE       // taille d'un nombre entier en bytes
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

size=INT_VALUE

[in] Le nombre de bytes (jusqu'à 4 inclus) qu'il faut lire. On prévoit les constantes correspondantes: CHAR_VALUE=1, SHORT_VALUE=2 et INT_VALUE=4, ainsi la fonction peut lire la valeur entière comme char, short ou int.

La valeur rendue

La valeur du type int. Il est nécessaire évidemment d'amener le résultat de cette fonction au type cible, c'est-à-dire à ce type de données, qu'il faut lire. Puisque revient la valeur comme int, on peut le transformer en n'importe quelle valeur entière. Le pointeur de fichier est déplacé par le nombre de bytes lus.

Note

A la lecture de moins 4 bytes, le résultat acquis sera toujours positif. Si un ou deux bytes sont lus, on peut exactement définir le signe du nombre par voie de la réduction évidente en conséquence vers le type char (1 byte) ou le type short (2 bytes). La réception du signe pour le nombre de trois bytes est non triviale, puisqu'il n'y a pas de type de basecorrespondant.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteInteger)

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Trends"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres pour la lecture des données
input string InpFileName="Trend.bin"; // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les variables globales
int      ind=0;
int      size=0;
datetime time_buff[];
//--- indicator buffers
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- allouons la mémoire pour le tableau
   ArrayResize(time_buff,def_size);
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- les variables auxiliaires
      int    arr_size;
      uchar  arr[];
      //--- lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //--- apprenons combien de symboles sont utilisés pour l'inscription du temps
         arr_size=FileReadInteger(file_handle,INT_VALUE);
         ArrayResize(arr,arr_size);
         for(int i=0;i<arr_size;i++)
            arr[i]=(char)FileReadInteger(file_handle,CHAR_VALUE);
         //--- retiendrons la valeur du temps
         time_buff[size]=StringToTime(CharArrayToString(arr));
         size++;
         //--- augmentons les tailles des tableaux s'ils sont remplis
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(time_buff,def_size);
           }
```

© 2000-2015, MetaQuotes Software Corp.

```
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- Le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- l'établissement des valeurs de l'indicateur qui ne seront pas visibles sur le g
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(close,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      buff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //---si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               //--- recevrons le prix
               buff[i]=close[i];
               //--- augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

© 2000-2015, MetaQuotes Software Corp.

IntegerToString, StringToInteger, Les types entiers, FileWriteInteger

© 2000-2015, MetaQuotes Software Corp.

FileReadLong

Lit le nombre entier du type long (8 bytes) de la position courante du fichier binaire.

```
long  FileReadLong(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type long.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteLong)

© 2000-2015, MetaQuotes Software Corp.

```
//+--//+------------------------------------------------------------------+
//|                                              Demo_FileReadLong.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Volume"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrYellow
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- les paramètres pour la lecture des données
input string InpFileName="Volume.bin"; // le nom du fichier
input string InpDirectoryName="Data";  // le nom du répertoire
//--- les variables globales
int      ind=0;
int      size=0;
long     volume_buff[];
datetime time_buff[];
//--- indicator buffers
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- Lisons d'abord combien de données sont dans le fichier
      size=(int)FileReadLong(file_handle);
      //--- allouons la mémoire pour les tableaux
      ArrayResize(volume_buff,size);
      ArrayResize(time_buff,size);
      //--- lisons des données du fichier
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadLong(file_handle);
         volume_buff[i]=FileReadLong(file_handle);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- le rattachement du tableau au tampon d'indicateur avec l'indice 0
   SetIndexBuffer(0,buff,INDICATOR_DATA);
```

© 2000-2015, MetaQuotes Software Corp.

```
//----l'établissement des valeurs de l'indicateur qui ne seront pas visibles sur le gr
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      buff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==time_buff[j])
              {
               buff[i]=(double)volume_buff[j];
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Les types entiers, FileReadInteger, FileWriteLong

© 2000-2015, MetaQuotes Software Corp.

FileReadNumber

Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type double.

```
double  FileReadNumber(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type double.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteString)

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadNumber.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- les paramètres pour la lecture des données
input string InpFileName="RSI.csv";   // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//--- les tampons d'indicateur
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- les variables du surachat
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- les variables de la survente
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- les variables des tailles des tableaux par défaut
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- allouons la mémoire pour les tableaux
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier : %s\\Files\\",TerminalInfoString(TERMINAL_
      double value;
      //--- lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //---lisons la première signification dans la chaîne
         value=FileReadNumber(file_handle);
         //--- lisons aux tableaux différents en fonction du résultat de la fonction
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
```

© 2000-2015, MetaQuotes Software Corp.

```
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- le rattachement des tableaux
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
//---- l'établissement des valeurs de l'indicateur, qui ne seront pas vus sur le graph
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| La lecture des données de la chaîne du fichier
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- lisons jusqu'à ce que nous arrivons à la fin d'une chaîne ou d'un fichier
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- rapprocherons le chariot, ayant lu le nombre
      if(flag)
         FileReadNumber(file_handle);
      //--- retenons la date courante
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- augmentons en cas de nécessité la taille du tableau
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- on a passé la première itération
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2015, MetaQuotes Software Corp.

```
{
```

© 2000-2015, MetaQuotes Software Corp.

```
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- le cycle pour les barres non-traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- la vérification s'il y a encore des données
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- si les dates correspondent, la barre se trouve dans la zone de surac
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - la couleur rouge
               color_buff[i]=0;
               //--- augmentons le compteur
               ovb_ind=j+1;
               break;
              }
           }
      //--- la vérification s'il y a encore des données
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- si les dates correspondent, la barre se trouve dans la zone de la su
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - la couleur bleue
               color_buff[i]=1;
               //---augmentons le compteur
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Le gestionnaire de l'événement ChartEvent                        |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
```

© 2000-2015, MetaQuotes Software Corp.

```
                     )
  {
//---changeons l'épaisseur de l'indicateur en fonction de l'échelle
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Voir aussi

FileWriteString

© 2000-2015, MetaQuotes Software Corp.

FileReadString

Lit du fichier la chaîne de la position courante de l'indicateur de fichier.

```
string  FileReadString(
   int  file_handle,     // handle du fichier
   int  length=-1        // longueur de la chaîne
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

length=-1

[in] Le nombre de caractères à lire.

La valeur rendue

La ligne lue (string).

Note

A la lecture du fichier bin il est obligatoire d'indiquer la longueur de la chaîne lue. A la lecture du fichier txt il ne faut pas indiquer la longueur de la chaîne, la chaîne sera lue de la position courante jusqu'au caractère de la conversion de la chaîne "\r\n". A la lecture du fichier csv il ne faut pas indiquer la longueur de la chaîne, la chaîne sera lue de la position actuelle jusqu'au délimiteur le plus proche ou jusqu'au caractère de la fin de la chaîne de texte.

Si le fichier est ouvert avec le drapeau FILE_ANSI,la ligne lue est transformée à Unicode.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteInteger)

© 2000-2015, MetaQuotes Software Corp.

```
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la lecture des données
input string InpFileName="Trend.bin"; // le nom du fichier
input string InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- les variables auxiliaires
      int    str_size;
      string str;
      //---lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //---apprenons combien de symboles sont utilisés pour l'inscription du temps
         str_size=FileReadInteger(file_handle,INT_VALUE);
         //--- lisons la chaîne
         str=FileReadString(file_handle,str_size);
         //--- imprimons la chaîne
         PrintFormat(str);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
  }
```

Voir aussi

Le type string, La conversion des données, FileWriteInteger

© 2000-2015, MetaQuotes Software Corp.

FileReadStruct

Lit du fichier binaire le contenu à la structure transmise à titre du paramètre. La lecture est produite de la position courante de l'indicateur de fichier.

```
uint  FileReadStruct(
   int           file_handle,        // handle du fichier
   const void&   struct_object,      // structure, où la lecture est faite
   int           size=-1             // dimension de la structure en bytes
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier du fichier binaire ouvert.

struct_object

[out] L'objet de la structure indiquée. La structure ne doit pas contenir les chaînes, les tableaux dynamiques et les fonctions virtuelles.

size=-1

[in] Le nombre de bytes, lesquels il faut lire. Si la dimension n'est pas spécifiée ou le nombre de bytes plus que la taille de la structure est indiqué, on utilise la dimension exacte de la structure indiquée.

La valeur rendue

En cas de succès la fonction rend le nombre de bytes lus. L'indicateur de fichier est déplacé par le même nombre de bytes.

Exemple (on utilise le fichier reçu à la suite du travail de l'exemple pour la fonction FileWriteStruct)

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Candles"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
#property indicator_separate_window
//--- les paramètres pour la réception des données
input string  InpFileName="EURUSD.txt"; // le nom du fichier
input string  InpDirectoryName="Data";  // le nom du répertoire
//+------------------------------------------------------------------+
//| La structure pour la sauvegarde des données de la bougie
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // le prix de l'ouverture
   double            close; // le prix de la clôture
   double            high;  // le prix maximum
   double            low;   // le prix minimum
   datetime          date;  // la date
  };
//--- les tampons d'indicateur
double       open_buff[];
double       close_buff[];
double       high_buff[];
double       low_buff[];
//--- les variables globales
candlesticks cand_buff[];
int          size=0;
int          ind=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int default_size=100;
   ArrayResize(cand_buff,default_size);
//--- ouvrons le fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_
      //--- lisons les données du fichier
      while(!FileIsEnding(file_handle))
        {
         //--- inscrirons les données au tableau
         FileReadStruct(file_handle,cand_buff[size]);
         size++;
```

© 2000-2015, MetaQuotes Software Corp.

```
         //--- vérifier un tableau sur l'encombrement
         if(size==default_size)
           {
            //---augmentons la dimension du tableau
            default_size+=100;
            ArrayResize(cand_buff,default_size);
           }
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont lues, le fichier %s est fermé",InpFileName);
     }
   else
     {
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
      return(INIT_FAILED);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
//--- une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
   ArraySetAsSeries(time,false);
//--- le cycle pour les bougies pas encore traitées
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- par défaut 0
      open_buff[i]=0;
      close_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      //---la vérification s'il y a encore des données
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- si les dates correspondent, utilisons la valeur du fichier
            if(time[i]==cand_buff[j].date)
              {
               open_buff[i]=cand_buff[j].open;
               close_buff[i]=cand_buff[j].close;
               high_buff[i]=cand_buff[j].high;
               low_buff[i]=cand_buff[j].low;
               //--- augmentons le compteur
               ind=j+1;
               break;
              }
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Les structures et les classes, FileWriteStruct

© 2000-2015, MetaQuotes Software Corp.

FileSeek

Déplace la position de l'indicateur de fichier par un nombre spécifié de bytes par rapport à la position indiquée.

```
bool  FileSeek(
   int                  file_handle,     // handle du fichier
   long                 offset,          // en bytes
   ENUM_FILE_POSITION   origin           // position pour le décompte
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

offset

[in] Le décalage dans les bytes (peut accepter et la valeur négative).

origin

[in] Le point de départ pour le décalage. Peut accepter une des valeurs de l'énumération ENUM_FILE_POSITION.

La valeur rendue

Rend true en cas du succès, autrement rend false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

Si le résultat de l'exécution de la fonction FileSeek() est le décalage négatif (la sortie pour "une gauche marge" le fichier), l'indicateur de fichier sera établi au début du fichier.

Si exposer la position pour "la marge droite" du fichier (plus que la taille du fichier), l'enregistrement ultérieur au fichier sera produite non de la fin du fichier, mais de la position exposée. En cela entre la fin précédente du fichier et la position exposée on inscrit les valeurs incertaines.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                                Demo_FileSeek.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="file.txt";     // le nom du fichier
input string InpDirectoryName="Data";    // le nom du répertoire
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- établissons la valeur de la variable pour la génération des nombres accidentels
   _RandomSeed=GetTickCount();
//--- les variables pour les positions du début des chaînes
   ulong pos[];
   int   size;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrons le fichier
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      //--- recevrons la position du début pour chaque chaîne dans le fichier
      GetStringPositions(file_handle,pos);
      //--- définissons le nombre de chaînes dans le fichier
      size=ArraySize(pos);
      if(!size)
        {
         //--- S'il n'y a pas de chaînes dans le fichier, terminons le travail
         PrintFormat("Le fichier %s est vide!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- choisirons le numéro de chaîne par hasard
      int ind=MathRand()%size;
      //--- déplaçons la position au début de cette chaîne
      if(FileSeek(file_handle,pos[ind],SEEK_SET)==true)
        {
      //--- lisons et imprimons la chaîne avec le numéro ind
         PrintFormat("Le texte de la chaîne avec le numéro %d: \"%s\"",ind,FileReadStr
        }
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Le fichier %s est fermé",InpFileName);
     }
   else
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
  }
//+------------------------------------------------------------------+
//| La fonction définit les positions du début pour chacune des chaînes dans le fichie
//| les place dans un tableau arr                                    |
//+------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
//--- la taille du tableau par défaut
   int def_size=127;
//--- allouons la mémoire pour le tableau
   ArrayResize(arr,def_size);
//--- le compteur des chaînes
   int i=0;
//--- s'il n'y a pas de la fin du fichier, il y a quand même une chaîne
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // le fichier est vide, sortons
//--- définissons le décalage en octets en fonction de l'encodage
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- trions les chaînes dans le cycle
   while(1)
     {
      //--- lisons la chaîne
      FileReadString(handle);
      //--- la vérification à la fin du fichier
      if(!FileIsEnding(handle))
        {
         //--- retiendrons la position de la chaîne suivante
         arr[i]=FileTell(handle)+shift;
         i++;
         //---augmentons la taille du tableau, s'il est rempli
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // la fin du fichier, sortons
     }
//---établissons la taille véritable du tableau
   ArrayResize(arr,i);
  }
```

© 2000-2015, MetaQuotes Software Corp.

FileSize

Rend la taille du fichier en bytes.

```
ulong  FileSize(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La valeur du type int.

Note

Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input ulong  InpThresholdSize=20;        // la frontière de la taille des fichiers en
input string InpBigFolderName="big";     // le dossier pour les grands fichiers
input string InpSmallFolderName="small"; // le dossier pour les petits fichiers
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // la variable pour la sauvegarde des noms des fichiers
   string   filter="*.csv"; // le filtre pour la recherche des fichiers
   ulong    file_size=0;    // la taille du fichier en octets
   int      size=0;         // le nombre de fichier
//--- imprimons la voie vers le dossier dans lequel on va travailler
   PrintFormat("Travaillons dans le dossier %s\\Files\\",TerminalInfoString(TERMINAL_C
//--- la réception du handle de la recherche dans la racine du dossier total de tous l
   long search_handle=FileFindFirst(filter,file_name,FILE_COMMON);
//--- vérifions si la fonction FileFindFirst() a travaillé avec succès
   if(search_handle!=INVALID_HANDLE)
     {
      //--- dans le cycle nous déplaçons les fichiers en fonction de leur taille
      do
        {
         //--- ouvrons le fichier
         ResetLastError();
         int file_handle=FileOpen(file_name,FILE_READ|FILE_CSV|FILE_COMMON);
         if(file_handle!=INVALID_HANDLE)
           {
            //--- recevrons la taille du fichier
            file_size=FileSize(file_handle);
            //--- fermons le fichier
            FileClose(file_handle);
           }
         else
           {
            PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreu
            continue;
           }
         //--- imprimons la taille du fichier
         PrintFormat("La taille du fichier %s est égal au %d octets",file_name,file_si
         //--- définirons la voie pour le déplacement du fichier
         string path;
         if(file_size>InpThresholdSize*1024)
            path=InpBigFolderName+"//"+file_name;
         else
            path=InpSmallFolderName+"//"+file_name;
         //--- déplaçons le fichier
         ResetLastError();
         if(FileMove(file_name,FILE_COMMON,path,FILE_REWRITE|FILE_COMMON))
            PrintFormat("Le fichier %s est déplacé",file_name);
         else
            PrintFormat("L'erreur, le code = %d",GetLastError());
        }
      while(FileFindNext(search_handle,file_name));
      //--- fermons le handle de la recherche
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
```

© 2000-2015, MetaQuotes Software Corp.

```
}
```

© 2000-2015, MetaQuotes Software Corp.

FileTell

Rend la position courante de l'indicateur de fichier du fichier correspondant ouvert.

```
ulong  FileTell(
   int  file_handle      // handle du fichier
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

La valeur rendue

La position courante du descripteur de fichier dans les bytes du début du fichier.

Note

Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                                Demo_FileTell.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string InpFileName="file.txt";     // le nom du fichier
input string InpDirectoryName="Data";    // le nom du répertoire
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- établissons la valeur de la variable pour la génération des nombres accidentels
   _RandomSeed=GetTickCount();
//--- les variables pour les positions du début des chaînes
   ulong pos[];
   int   size;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- ouvrons le fichier
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour la lecture",InpFileName);
      //--- recevrons la position du début pour chaque chaîne dans le fichier
      GetStringPositions(file_handle,pos);
      //--- définissons le nombre de chaînes dans le fichier
      size=ArraySize(pos);
      if(!size)
        {
         //--- S'il n'y a pas de chaînes dans le fichier, terminons le travail
         PrintFormat("Le fichier %s est vide!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- choisirons le numéro de chaîne par hasard
      int ind=MathRand()%size;
      //--- déplaçons la position au début de cette chaîne
      FileSeek(file_handle,pos[ind],SEEK_SET);
      //--- lisons et imprimons la chaîne avec le numéro ind
      PrintFormat("Le texte de la chaîne avec le numéro %d: \"%s\"",ind,FileReadStrinc
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Le fichier %s est fermé",InpFileName);
     }
   else
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
  }
//+------------------------------------------------------------------+
//| La fonction définit les positions du début pour chacune des chaînes dans le fichie
//| les place dans un tableau arr                                    |
//+------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- la taille du tableau par défaut
   int def_size=127;
//--- allouons la mémoire pour le tableau
   ArrayResize(arr,def_size);
//--- le compteur des chaînes
   int i=0;
//--- s'il n'y a pas de la fin du fichier, il y a quand même une chaîne
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // le fichier est vide, sortons
//--- définissons le décalage en octets en fonction de l'encodage
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- trions les chaînes dans le cycle
   while(1)
     {
      //--- lisons la chaîne
      FileReadString(handle);
      //--- la vérification à la fin du fichier
      if(!FileIsEnding(handle))
        {
         //--- retiendrons la position de la chaîne suivante
         arr[i]=FileTell(handle)+shift;
         i++;
         //---augmentons la taille du tableau, s'il est rempli
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // la fin du fichier, sortons
     }
//---établissons la taille véritable du tableau
   ArrayResize(arr,i);
  }
```

© 2000-2015, MetaQuotes Software Corp.

FileWrite

Inscrit les données au fichier comme CSV ou TXT, le délimiteur entre les données est inséré automatiquement, s'il n'est pas égal à 0. Après avoir écrit dans le fichier, le caractère de la fin de la ligne "\r\n" sera ajouté.

```
uint  FileWrite(
   int  file_handle,     // handle du fichier
   ...                   // liste des paramètres inscrits
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

...

[in] La liste des paramètres divisés par les virgules, pour l'enregistrement au fichier. Le nombre de paramètres entrés au fichier ne doit pas excéder 63.

La valeur rendue

Le nombre de bytes écrits.

Note

A la sortie les données numériques seront transformées au format de texte (regarde la fonction Print()). Les données du type double sont sorties à près de 16 chiffres décimaux après le point, en cela les données peuvent être sortis aux formats traditionnel ou scientifique – en fonction de que l'enregistrement sera plus compact. Les données du type float sont sorties avec 5 chiffres décimaux après le point. Pour la sortie des nombres matériels avec une autre exactitude ou dans le format évidemment indiqué il est nécessaire d'utiliser la fonction DoubleToString().

Les nombres du type bool sont sorties en forme des lignes "true" ou "false". Les nombres du type datetime sont sorties au format "YYYY.MM.DD HH:MI:SS".

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                                Demo_FileWrite.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string             InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // le temps trame
input int                InpFastEMAPeriod=12;              // la période de la moyenne
input int                InpSlowEMAPeriod=26;              // la période de la moyenne
input int                InpSignalPeriod=9;                // la période de la prise d
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // le type du prix
input datetime           InpDateStart=D'2012.01.01 00:00'; // la date de commencement
//--- les paramètres pour l'enregistrement des données au fichier
input string             InpFileName="MACD.csv";  // le nom du fichier
input string             InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // la date de la fin du copiage des données
   bool     sign_buff[]; // le tableau des signaux (true - l'achat, false - la vente)
   datetime time_buff[]; // le tableau du temps de l'arrivée des signaux
   int      sign_size=0; // la taille des tableaux des signaux
   double   macd_buff[]; // le tableau des valeurs de l'indicateur
   datetime date_buff[]; // le tableau des dates de l'indicateur
   int      macd_size=0; // la taille des tableaux de l'indicateur
//--- le temps de la fin - courant
   date_finish=TimeCurrent();
//--- recevrons le handle de l'indicateur MACD
   ResetLastError();
   int macd_handle=iMACD(InpSymbolName,InpSymbolPeriod,InpFastEMAPeriod,InpSlowEMAPeri
   if(macd_handle==INVALID_HANDLE)
     {
      //--- on n'a pas réussi de recevoir le hadle de l'indicateur
      PrintFormat("L'erreur de la réception du handle de l'indicateur. Le code de l'er
      return;
     }
//--- on se trouve dans le cycle, pendant que l'indicateur ne compte pas toutes les va
   while(BarsCalculated(macd_handle)==-1)
      Sleep(10); // le retard pour que l'indicateur puisse de calculer les valeurs
//---copions les valeurs de l'indicateur pour la période définie
   ResetLastError();
   if(CopyBuffer(macd_handle,0,InpDateStart,date_finish,macd_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs de l'indicateur. Le code de
      return;
     }
//--- copions le temps correspondant pour les valeurs de l'indicateur
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps Le code de l'erreu
      return;
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- désallouons la mémoire occupée par l'indicateur
   IndicatorRelease(macd_handle);
//---recevrons la taille du tampon
   macd_size=ArraySize(macd_buff);
//--- analyserons les données et sauvegardons les signaux de l'indicateur aux tableaux
   ArrayResize(sign_buff,macd_size-1);
   ArrayResize(time_buff,macd_size-1);
   for(int i=1;i<macd_size;i++)
     {
      //--- le signal à l'achat
      if(macd_buff[i-1]<0 && macd_buff[i]>=0)
        {
         sign_buff[sign_size]=true;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
      //--- le signal à la vente
      if(macd_buff[i-1]>0 && macd_buff[i]<=0)
        {
         sign_buff[sign_size]=false;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
     }
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_
      //--- d'abord enregistrons la quantité de signaux
      FileWrite(file_handle,sign_size);
      //--- enregistrons le temps des signaux et leurs valeurs au fichier
      for(int i=0;i<sign_size;i++)
         FileWrite(file_handle,time_buff[i],sign_buff[i]);
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont enregistrés, le fichier %s est fermé",InpFileName)
     }
   else
      PrintFormat("On n'a pas réussi d'ouvrir le fichier %s, Le code de l'erreur = %d'
  }
```

Voir aussi

Comment, Print, StringFormat

© 2000-2015, MetaQuotes Software Corp.

FileWriteArray

Inscrit au fichier du type BIN les tableaux des tous les types, sauf les tableaux de chaîne (peut être un tableau des structures ne contenant pas les chaînes et les tableaux dynamiques).

```
uint  FileWriteArray(
   int           file_handle,          // handle du fichier
   const void&   array[],              // tableau
   int           start=0,              // index initial dans le tableau
   int           count=WHOLE_ARRAY     // nombre d'éléments
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

array[]

[out] Le tableau pour l'enregistrement.

start=0

[in] L'index initial dans le tableau (le numéro du premier élément inscrit).

count=WHOLE_ARRAY

[in] Le nombre d'éléments inscrits (WHOLE_ARRAY signifie que s'inscrivent tous les éléments à partir du numéro start jusqu'à la fin du tableau).

La valeur rendue

Le nombre d'éléments inscrits.

Note

Le tableau de chaîne peut s'inscrire seulement au fichier du type TXT. Dans ce cas les chaînes s'achèvent automatiquement par les caractères de la fin de la ligne "\r\n". En fonction du type du fichier ANSI ou UNICODE, les chaînes se transforment au codage ansi, ou non.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteArray.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- les paramètres d'entrée
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| La structure pour stocker des données sur les prix                                 |
//+------------------------------------------------------------------+
struct prices
  {
   datetime          date; // lq date
   double            bid;  // le prix bid
   double            ask;  // le prix ask
  };
//--- les variables globales
int    count=0;
int    size=20;
string path=InpDirectoryName+"//"+InpFileName;
prices arr[];
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- l'allocation de la mémoire pour un tableau
   ArrayResize(arr,size);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- un enregistrement des chaînes count restées, si count<n
   WriteData(count);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- sauvegardons les données au tableau
   arr[count].date=TimeCurrent();
   arr[count].bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   arr[count].ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- affichons les données courantes
   Print("Date = ",arr[count].date," Bid = ",arr[count].bid," Ask = ",arr[count].ask);
//---augmentons le compteur
   count++;
//--- si le tableau est rempli, enregistrons les données dans un fichier, et le remett
   if(count==size)
     {
      WriteData(size);
      count=0;
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   }
//+------------------------------------------------------------------+
//| L'enregistrement n des éléments d'un tableau dans un fichier
//+------------------------------------------------------------------+
void WriteData(const int n)
   {
//--- ouvrons le fichier
    ResetLastError();
    int handle=FileOpen(path,FILE_READ|FILE_WRITE|FILE_BIN);
    if(handle!=INVALID_HANDLE)
      {
       //--- enregistrons les données du tableau à la fin du fichier
       FileSeek(handle,0,SEEK_END);
       FileWriteArray(handle,arr,0,n);
       //--- fermons le fichier
       FileClose(handle);
      }
    else
       Print("Failed to open the file, error ",GetLastError());
   }
```

Voir aussi

Les variables, FileSeek

© 2000-2015, MetaQuotes Software Corp.

FileWriteDouble

Inscrit au fichier la valeur du type double de la position courante de l'indicateur de fichier.

```
uint  FileWriteDouble(
   int     file_handle,     // handle du fichier
   double  value            // valeur pour l'enregistrement
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

value

[in] La valeur du type double.

La valeur rendue

En cas du succès la fonction rend le nombre inscrit des bytes (dans ce cas sizeof(double)=8). L'indicateur de fichier se déplace sur le même nombre de bytes.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                        Demo_FileWriteDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//---  montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string             InpSymbolName="EURJPY";           // la paire de devise
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_M15;       // le temps trame
input int                InpMAPeriod=10;                   // la période du lissage
input int                InpMAShift=0;                     // le déplacement de l'indi
input ENUM_MA_METHOD     InpMAMethod=MODE_SMA;             // le type du lissage
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // le type du prix
input datetime           InpDateStart=D'2013.01.01 00:00'; // la date de commencement
//--- les paramètres pour l'enregistrement des données au fichier
input string             InpFileName="MA.csv";     // le nom du fichier
input string             InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   ma_buff[];
   datetime time_buff[];
   int      size;
//--- recevrons le handle de l'indicateur MA
   ResetLastError();
   int ma_handle=iMA(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpMAShift,InpMAMethod,
   if(ma_handle==INVALID_HANDLE)
     {
      //--- on n'a pas réussi de recevoir le hadle de l'indicateur
      PrintFormat("L'erreur de la réception du handle de l'indicateur. Le code de l'er
      return;
     }
//--- on se trouve dans le cycle, pendant que l'indicateur ne compte pas toutes ses va
   while(BarsCalculated(ma_handle)==-1)
      Sleep(20); // le retard pour que l'indicateur puisse de calculer ses valeurs
   PrintFormat("Les valeurs de l'indicateur seront enregistrées au fichier, à partir d
//--- copions les valeurs de l'indicateur
   ResetLastError();
   if(CopyBuffer(ma_handle,0,InpDateStart,date_finish,ma_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs de l'indicateur. Le code de
      return;
     }
//--- copions le temps de l'apparition des barres correspondantes
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//---recevrons la taille du tampon
   size=ArraySize(ma_buff);
//--- désallouons la mémoire occupée par l'indicateur
   IndicatorRelease(ma_handle);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_
      //--- d'abord enregistrons la taille de l'extrait des données
      FileWriteDouble(file_handle,(double)size);
      //--- enregistrons le temps et les valeurs de l'indicateur dans le fichier
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteDouble(file_handle,ma_buff[i]);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sonr enregistrés, le fichier %s est fermé",InpFileName)
     }
   else
      PrintFormat("On n'a pas réussi d'ouvrir le fichier %s, Le code de l'erreur = %d'
  }
```

Voir aussi

Les types entiers (double, float)

© 2000-2015, MetaQuotes Software Corp.

FileWriteFloat

Inscrit au fichier bin la valeur du paramètre du type float de la position courante de l'indicateur de fichier.

```
uint  FileWriteFloat(
   int    file_handle,     // handle du fichier
   float  value            // valeur enregistrée
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

value

[in] La valeur du type float.

La valeur rendue

En cas du succès la fonction rend le nombre inscrit des bytes (dans ce cas sizeof(float)=4). L'indicateur de fichier se déplace sur le même nombre de bytes.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string          InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_M15;       // le temps trame
input datetime        InpDateStart=D'2013.01.01 00:00'; // la date de commencement du
//--- les paramètres pour l'enregistrement des données au fichier
input string          InpFileName="Close.bin"; // le nom du fichier
input string          InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- copions le prix de la clôture pour chaque barre
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("On n'a pas réussi de copier les valeurs des prix de la clôture. Le
      return;
     }
//---copions le temps pour chaque barre
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//--- recevrons la taille du tampon
   size=ArraySize(close_buff);
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- enregistrons le temps et les valeurs des prix de la clôture au fichier
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteFloat(file_handle,(float)close_buff[i]);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont enregistrés, le fichier %s est fermé",InpFileName)
     }
   else
      PrintFormat("On n'a pas réussi d'ouvrir le fichier %s, Le code de l'erreur = %d'
```

© 2000-2015, MetaQuotes Software Corp.

```
}
```

Voir aussi

Les types entiers (double, float), FileWriteDouble

© 2000-2015, MetaQuotes Software Corp.

FileWriteInteger

Inscrit au fichier bin la valeur du paramètre du type int de la position courante de l'indicateur de fichier.

```
uint  FileWriteInteger(
   int  file_handle,         // handle du fichier
   int  value,               // valeur enregistrée
   int  size=INT_VALUE       // taille en bytes
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

value

[in] La valeur entière.

size=INT_VALUE

[in] Le nombre de bytes (jusqu'à 4 y compris), qu'il faut enregistrer. On prévoit les constantes correspondantes: CHAR_VALUE=1, SHORT_VALUE=2 et INT_VALUE=4, ainsi la fonction peut enregistrer la valeur entière du type char, uchar, short, ushort, int ou uint

La valeur rendue

En cas du succès la fonction rend le nombre inscrit des bytes. L'indicateur de fichier se déplace sur le même nombre de bytes.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                        Demo_FileWriteInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string             InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // le temps trame
input datetime           InpDateStart=D'2013.01.01 00:00'; // la date de commencement
//--- les paramètres pour l'enregistrement des données au fichier
input string             InpFileName="Trend.bin"; // le nom du fichier
input string             InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
//---oblitérons la valeur de l'erreur
   ResetLastError();
//--- copions le prix de la clôture pour chaque barre
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("On n'a pas réussi de copier les valeurs des prix de la clôture. Le
      return;
     }
//---copions le temps pour chaque barre
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//--- recevrons la taille du tampon
   size=ArraySize(close_buff);
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("Le voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //---
      int   up_down=0; // le drapeau de la tendance
      int   arr_size;  // la taille du tableau arr
      uchar arr[];     // le tableau du type uchar
      //--- enregistrons les valeurs du temps au fichier
      for(int i=0;i<size-1;i++)
        {
         //--- comparerons les prix de la clôture des barres courante et suivante
         if(close_buff[i]<=close_buff[i+1])
           {
            if(up_down!=1)
              {
```

© 2000-2015, MetaQuotes Software Corp.

```
                //--- enregistrons la valeur de la date au fichier, en utilisant FileWr
                StringToCharArray(TimeToString(time_buff[i]),arr);
                arr_size=ArraySize(arr);
                //--- d'abord enregistrons le nombre de symboles dans le tableau
                FileWriteInteger(file_handle,arr_size,INT_VALUE);
                //--- enregistrons les symboles eux-mêmes
                for(int j=0;j<arr_size;j++)
                   FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
                //--- changeons le drapeau de la tendance
                up_down=1;
               }
            }
          else
            {
             if(up_down!=-1)
               {
                //--- enregistrons la valeur de la date au fichier, en utilisant FileWr
                StringToCharArray(TimeToString(time_buff[i]),arr);
                arr_size=ArraySize(arr);
                //---d'abord enregistrons le nombre de symboles dans le tableau
                FileWriteInteger(file_handle,arr_size,INT_VALUE);
                //--- enregistrons les symboles eux-mêmes
                for(int j=0;j<arr_size;j++)
                   FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
                //--- changeons le drapeau de la tendance
                up_down=-1;
               }
            }
         }
       //--- fermons le fichier
       FileClose(file_handle);
       PrintFormat("Les données sont enregistrés, le fichier %s est fermé",InpFileName)
      }
    else
       PrintFormat("On n'a pas réussi d'ouvrir le fichier %s, Le code de l'erreur = %d'
  }
```

Voir aussi

IntegerToString, StringToInteger, Les types entiers

© 2000-2015, MetaQuotes Software Corp.

FileWriteLong

Inscrit au fichier bin la valeur du paramètre du type long de la position courante de l'indicateur de fichier.

```
uint  FileWriteLong(
   int   file_handle,     // handle du fichier
   long  value            // valeur enregistrée
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

value

[in] La valeur du type long.

La valeur rendue

En cas du succès la fonction rend le nombre inscrit des bytes (dans ce cas sizeof(long)=8). L'indicateur de fichier se déplace sur le même nombre de bytes.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteLong.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string          InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // le temps trame
input datetime        InpDateStart=D'2013.01.01 00:00'; // la date de commencement du
//--- les paramètres pour l'enregistrement des données au fichier
input string          InpFileName="Volume.bin"; // le nom du fichier
input string          InpDirectoryName="Data";  // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   long     volume_buff[];
   datetime time_buff[];
   int      size;
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- copions les volumes de tick pour chaque barre
   if(CopyTickVolume(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,volume_buf
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du volume de tick. Le code
      return;
     }
//---copions le temps pour chaque barre
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//--- recevrons la taille du tampon
   size=ArraySize(volume_buff);
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_I
      //--- d'abord enregistrons la taille de l'extrait des données
      FileWriteLong(file_handle,(long)size);
      //--- enregistrons le temps et les valeurs du volume au fichier
      for(int i=0;i<size;i++)
        {
         FileWriteLong(file_handle,(long)time_buff[i]);
         FileWriteLong(file_handle,volume_buff[i]);
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont enregistrés, le fichier %s est fermé",InpFileName)
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   else
      PrintFormat("On n'a pas réussi d'ouvrir le fichier %s, Le code de l'erreur = %d'
}
```

Voir aussi

Les types entiers, FileWriteInteger

© 2000-2015, MetaQuotes Software Corp.

FileWriteString

Inscrit au fichier comme BIN, CSV ou TXT la valeur du paramètre du type string de la position courante du pointeur de fichier. A l'enregistrement au fichier comme CSV ou TXT, si dans la ligne il y a un symbole '\n' (LF) sans symbole précédant '\r' (CR), devant le symbole '\n' on écrit le symbole manquant '\r'.

```
uint  FileWriteString(
   int           file_handle,     // handle du fichier
   const string  text_string,     // chaîne enregistrée
   int           length=-1        // nombre de symboles
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

text_string

[in] La chaîne.

length=-1

[in] Le nombre de caractères, qu'il faut enregistrer. Le paramètre est nécessaire à l'enregistrement de la chaîne au fichier du type BIN. Si la taille n'est pas indiqué, s'inscrit toute la chaîne sans le 0 terminant. Si la taille indiquée est plus petite que la longueur de la chaîne, on s'inscrit toute la chaîne sans le 0 terminant. Si on indique la taille plus que la longueur de la chaîne, la chaîne s'enregistre par le nombre correspondant de zéros. Pour les fichiers comme CSV et TXT ce paramètre est ignoré et la ligne est enregistrée entièrement.

La valeur rendue

En cas du succès la fonction rend le nombre inscrit des bytes. L'indicateur de fichier se déplace sur le même nombre de bytes.

Note

Il faut avoir en vue que pendant l'enregistrement au fichier ouvert avec le drapeau FILE_UNICODE (ou sans le drapeau FILE_ANSI), le nombre inscrit de bytes sera au 2 fois plus de nombre de caractères inscrits de la chaîne. A l'enregistrement au fichier ouvert avec le drapeau FILE_ANSI, le nombre de bytes enregistrés coïncidera avec le nombre de caractères inscrits de la chaîne.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                        Demo_FileWriteString.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string             InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // le temps trame
input int                InpMAPeriod=14;                   // la période de la moyenne
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // le type du prix
input datetime           InpDateStart=D'2013.01.01 00:00'; // la date du commencement
//--- les paramètres pour l'enregistrement des données au fichier
input string             InpFileName="RSI.csv";    // le nom du fichier
input string             InpDirectoryName="Data"; // le nom du répertoire
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // la date de la fin du copiage des données
   double   rsi_buff[];  // le tableau des valeurs de l'indicateur
   datetime date_buff[]; // le tableau des dates de l'indicateur
   int      rsi_size=0;  // la taille des tableaux de l'indicateur
//--- le temps de la fin - courant
   date_finish=TimeCurrent();
//--- recevrons le handle de l'indicateur RSI
   ResetLastError();
   int rsi_handle=iRSI(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- on n'a pas réussi de recevoir le hadle de l'indicateur
      PrintFormat("L'erreur de la réception du handle de l'indicateur. Le code de l'er
      return;
     }
//---on se trouve dans le cycle, pendant que l'indicateur ne compte pas toutes ses val
   while(BarsCalculated(rsi_handle)==-1)
      Sleep(10); // le retard pour que l'indicateur puisse de calculer ses valeurs
//--- copions les valeurs de l'indicateur pour la période définie
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,date_finish,rsi_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs de l'indicateur. Le code de
      return;
     }
//--- copions le temps correspondant pour les valeurs de l'indicateur
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//---désallouons la mémoire occupée par l'indicateur
   IndicatorRelease(rsi_handle);
//--- recevrons la taille du tampon
   rsi_size=ArraySize(rsi_buff);
//--- ouvrirons le fichier pour l'enregistrement des valeurs de l'indicateur (s'il n'e
```

© 2000-2015, MetaQuotes Software Corp.

```
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_
      //--- préparerons les variables auxiliaires
      string str="";
      bool   is_formed=false;
      //--- écrirons les dates sur les zones de surachat et de survente
      for(int i=0;i<rsi_size;i++)
        {
         //--- la vérification des valeurs de l'indicateur
         if(rsi_buff[i]>=70 || rsi_buff[i]<=30)
           {
            //--- si c'est la première valeur dans cette zone
            if(!is_formed)
              {
               //--- ajoutons la valeur et la date
               str=(string)rsi_buff[i]+"\t"+(string)date_buff[i];
               is_formed=true;
              }
            else
               str+="\t"+(string)rsi_buff[i]+"\t"+(string)date_buff[i];
            //--- le passage à l'itération suivante du cycle
            continue;
           }
         //--- la vérification du drapeau
         if(is_formed)
           {
            //---la ligne est formée, enregistrons au fichier
            FileWriteString(file_handle,str+"\r\n");
            is_formed=false;
           }
        }
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont enregistrés, le fichier %s est fermé",InpFileName)
     }
   else
      PrintFormat("Les données sont inscrites, le fichier %s est fermé = %d",InpFileNa
  }
```

Voir aussi

Le type string, StringFormat

© 2000-2015, MetaQuotes Software Corp.

FileWriteStruct

Inscrit au fichier bin le contenu de la structure, transmise à titre du paramètre de la position courante de l'indicateur de fichier.

```
uint  FileWriteStruct(
   int           file_handle,       // handle du fichier
   const void&   struct_object,     // lien à l'objet
   int           size=-1            // taille pour enregistrement en bytes
   );
```

Paramètres

file_handle

[in] Le descripteur de fichier rendu par la fonction FileOpen().

struct_object

[in] Le lien à l'objet de la structure indiquée. La structure ne doit pas contenir les chaînes, les tableaux dynamiques et les fonctions virtuelles.

size=-1

[in] Le nombre de bytes, lesquels il faut lire. Si la dimension n'est pas spécifiée ou le nombre de bytes plus que la taille de la structure est indiqué, on enregistre toute la structure.

La valeur rendue

En cas de succès la fonction rend le nombre de bytes lus. L'indicateur de fichier est déplacé par le même nombre de bytes.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWiteStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres pour la réception des données du terminal
input string          InpSymbolName="EURUSD";           // la paire de devise
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // le temps trame
input datetime        InpDateStart=D'2013.01.01 00:00'; // la date de commencement du
//--- les paramètres pour l'enregistrement des données au fichier
input string          InpFileName="EURUSD.txt";         // le nom du fichier
input string          InpDirectoryName="Data";          // le nom du répertoire
//+------------------------------------------------------------------+
//| La structure pour stocker des données de la bougie
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // le prix de l'ouverture
   double            close; // le prix de la clôture
   double            high;  // le prix maximum
   double            low;   // le prix minimum
   datetime          date;  // la date
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime     date_finish=TimeCurrent();
   int          size;
   datetime     time_buff[];
   double       open_buff[];
   double       close_buff[];
   double       high_buff[];
   double       low_buff[];
   candlesticks cand_buff[];
//--- oblitérons la valeur de l'erreur
   ResetLastError();
//--- recevrons le temps de l'apparition des barres de la gamme
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs du temps. Le code de l'erre
      return;
     }
//--- recevrons les prix maximals des barres de la gamme
   if(CopyHigh(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,high_buff)==-1)
     {
      PrintFormat(On n'a pas réussi de copier les valeurs des prix maximales. Le code
      return;
     }
//--- recevrons les prix minimales des barres de la gamme
   if(CopyLow(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,low_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs des prix minimales. Le code
      return;
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- recevrons les prix de l'ouverture des barres de la gamme
   if(CopyOpen(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,open_buff)==-1)
     {
      PrintFormat("On n'a pas réussi de copier les valeurs des prix de l'ouverture. Le
      return;
     }
//--- recevrons les prix de la clôture des barres de la gamme
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("On n'a pas réussi de copier les valeurs des prix de la clôture. Le
      return;
     }
//--- définirons la dimension des tableaux
   size=ArraySize(time_buff);
//--- sauvegardons toutes les structures données dans le tableau
   ArrayResize(cand_buff,size);
   for(int i=0;i<size;i++)
     {
      cand_buff[i].open=open_buff[i];
      cand_buff[i].close=close_buff[i];
      cand_buff[i].high=high_buff[i];
      cand_buff[i].low=low_buff[i];
      cand_buff[i].date=time_buff[i];
     }

//--- ouvrirons le fichier pour l'enregistrement du tableau de la structure au fichier
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("Le fichier %s est ouvert pour l'enregistrement",InpFileName);
      PrintFormat("La voie vers le fichier: %s\\Files\\",TerminalInfoString(TERMINAL_
      //--- préparerons le compteur de la quantité des octets
      uint counter=0;
      //---enregistrons les valeurs du tableau dans le cycle
      for(int i=0;i<size;i++)
         counter+=FileWriteStruct(file_handle,cand_buff[i]);
      PrintFormat(" %d octets de l'information sont enregistrés au fichier %s",InpFile
      PrintFormat("Au total octets: %d * %d * %d = %d, %s",size,5,8,size*5*8,size*5*8=
      //--- fermons le fichier
      FileClose(file_handle);
      PrintFormat("Les données sont enregistrées, le fichier %s est fermé",InpFileName
     }
   else
      PrintFormat("On n' a pas réussi à ouvrir le fichier %s, le code de l'erreur = %d
  }
```

Voir aussi

Les structures et les classes

© 2000-2015, MetaQuotes Software Corp.

FolderCreate

Crée le répertoire dans le dossier Files (en fonction de la valeur common_flag)

```
bool  FolderCreate(
   string  folder_name,       // chaîne avec le nom du dossier créé
   int     common_flag=0      // domaine d'action
   );
```

Paramètres

folder_name

[in] Le nom du répertoire, qu'il faut créer. Contient le chemin complet vers le dossier.

common_flag=0

[in] Le drapeau, définissant la situation du répertoire. Si common_flag=FILE_COMMON, le répertoire se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common \Files. Dans le cas contraire le répertoire se trouve dans le dossier local (MQL5\Files ou MQL5 \Tester\Files en cas du test).

La valeur rendue

Rend true en cas du succès, autrement rend false.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                              Demo_FolderCreate.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- la description
#property description "Le script montre l'example de l'utilisation de FolderCreate()."
#property description "Le paramètre externe spécifie le répertoire pour la création de
#property description "Après avoir exécuté le script la structure de dossiers sera cré

//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- un paramètre d'entrée définit le dossier où le script fonctionne
input bool     common_folder=false; // le dossier commun de tous les terminaux
int            flag=0;              // les valeurs du drapeau définissent la place de
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string working_folder;
//---définissons la valeur du drapeau, si un paramètre externe common_folder==true
   if(common_folder)
     {
      flag=FILE_COMMON;
      //--- trouvons le dossier, dans lequel nous travaillons
      working_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH)+"\\MQL5\\Files";
     }
   else working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- le classeur, que nous créons dans le classeur MQL5\Files
   string root="Folder_A";
   if(CreateFolder(working_folder,root,flag))
     {
      //--- y créons un dossier enfant Child_Folder_B1
      string folder_B1="Child_Folder_B1";
      string path=root+"\\"+folder_B1;          // créons le nom du classeur en tenant
      if(CreateFolder(working_folder,path,flag))
        {
         //--- dans ce dossier nous créons encore 3 dossiers enfant
         string folder_C11="Child_Folder_C11";
         string child_path=path+"\\"+folder_C11;// créons le nom du dossier en tenant
         CreateFolder(working_folder,child_path,flag);
         //--- le deuxième dossier enfant
         string folder_C12="Child_Folder_C12";
         child_path=path+"\\"+folder_C12;
         CreateFolder(working_folder,child_path,flag);

         //--- le troisième dossier enfant
         string folder_C13="Child_Folder_C13";
         child_path=path+"\\"+folder_C13;
         CreateFolder(working_folder,child_path,flag);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Tente de créer le dossier et déduit les messages                 |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool CreateFolder(string working_folder,string folder_path,int file_flag)
  {
//--- le message de débogage
   PrintFormat("folder_path=%s",folder_path);
//--- la tentative de créer le dossier par rapport au chemin MQL5\Files
   if(FolderCreate(folder_path,file_flag))
     {
      //--- déduisons le chemin complet jusqu'au dossier créé
      PrintFormat("On a créé le dossier %s",working_folder+"\\"+folder_path);
      //--- oblitérons le code de l'erreur
      ResetLastError();
      //--- rendons le succès
      return true;
     }
   else
      PrintFormat("On n'a pas réussi à créer le dossier %s. Le code de l'erreur %d",wo
//--- un échec
   return false;
  }
```

Voir aussi

FileOpen(), FolderClean(), FileCopy()

© 2000-2015, MetaQuotes Software Corp.

FolderDelete

Supprime le répertoire indiqué. Si le dossier n'est pas vide, elle ne peut pas être supprimé.

```
bool  FolderDelete(
   string  folder_name,       // chaîne avec le nom du dossier supprimé
   int     common_flag=0      // domaine d'action
   );
```

Paramètres

folder_name

[in] Le nom du répertoire, qu'il faut supprimer. Contient le chemin complet vers le dossier.

common_flag=0

[in] Le drapeau, définissant la situation du répertoire. Si common_flag=FILE_COMMON, le répertoire se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common \Files. Dans le cas contraire le répertoire se trouve dans le dossier local (MQL5\files ou MQL5 \tester\files en cas du test).

La valeur rendue

Rend true en cas du succès, autrement rend false.

Note

Si le répertoire contient au moins un fichier et/ou le sous-répertoire, l'effacement d'un tel répertoire est impossible, il est nécessaire préalablement de l'effacer. Le nettoyage total du dossier de tous les fichiers et les sous-dossiers inclus s'effectue à l'aide de la fonction FolderClean().

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                              Demo_FolderDelete.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- la description
#property description "Le script montre un exemple de l'utilisation FolderDelete()."
#property description "D'abord, deux dossiers sont créés, l'un est vide et l'autre cor
#property description "Si vous essayez de supprimer un dossier non vide vous recevrez

//--- montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string   firstFolder="empty";     // un dossier vide
input string   secondFolder="nonempty";// le dossier, dans lequel il y aura un fichier
string filename="delete_me.txt";       // le nom du fichier que nous créons dans le do
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- écrivons ici le handle du fichier
   int handle;
//---savons dans quel dossier nous travaillons
   string working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- le message de débogage
   PrintFormat("working_folder=%s",working_folder);
//--- la tentative de créer le dossier vide par rapport au chemin MQL5\Files
   if(FolderCreate(firstFolder,0)) // 0 signifie que nous travaillons dans un dossier
     {
      //--- déduisons un chemin complet jusqu'au dossier créé
      PrintFormat("On a créé le dossier %s",working_folder+"\\"+firstFolder);
      //--- oblitérons le code de l'erreur
      ResetLastError();
     }
   else
      PrintFormat("On n'a pas réussi à créer le dossier %s. Le code de l'erreur %d",wo

//--- maintenant nous créons le dossier non vide à l'aide de la fonction FileOpen()
   string filepath=secondFolder+"\\"+filename;  // formeront le chemin vers le fichier
   handle=FileOpen(filepath,FILE_WRITE|FILE_TXT); // le drapeau FILE_WRITE dans ce cas
   if(handle!=INVALID_HANDLE)
      PrintFormat("on a ouvert le fichier pour la lecture %s",working_folder+"\\"+file
   else
      PrintFormat("On n'a pas réussi à créer le fichier %s dans le dossier %s. Le code

   Comment(StringFormat("préparons de supprimer les dossiers %s et %s", firstFolder, s
//--- Une petite pause à 5 secondes pour que nous puissions lire le message sur le gra
   Sleep(5000); // Sleep() on ne peut pas utiliser dans les indicateurs!

//--- déduisons la boîte de dialogue et nous demandons à l'utilisateur
   int choice=MessageBox(StringFormat("Supprimer les dossiers %s et %s?", firstFolder,
                         "La suppression des dossiers",
                         MB_YESNO|MB_ICONQUESTION); //  Il y aura deux boutons - "Yes"

//--- effectuons des actions selon l'option choisie
   if(choice==IDYES)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- effaçons le commentaire sur le graphique
      Comment("");
      //--- déduisons le message au journal "Experts"
      PrintFormat("Essayons de supprimer les dossiers %s et %s",firstFolder, secondFol
      ResetLastError();
      //--- supprimons un dossier vide
      if(FolderDelete(firstFolder))
         //--- doivent voir ce message, puisque le dossier est vide
         PrintFormat("Le dossier %s est supprimé avec succès",firstFolder);
      else
         PrintFormat("On n'a pas réussi à supprimer le dossier %s. Le code de l'erreu

      ResetLastError();
      //--- supprimons le dossier qui contient le fichier
      if(FolderDelete(secondFolder))
         PrintFormat("Le dossier %s est supprimé avec succès", secondFolder);
      else
         //--- doivent voir ce message, puisque dans le dossier il y a un fichier
         PrintFormat("On n'a pas réussi à supprimer le dossier %s. Le code de l'erreu
     }
   else
      Print("La suppression est annulée");
//---
  }
```

Voir aussi

FileOpen(), FolderClean(), FileMove()

© 2000-2015, MetaQuotes Software Corp.

FolderClean

Supprime tous les fichiers dans le dossier indiqué.

```
bool  FolderClean(
   string  folder_name,       // chaîne avec le nom du dossier
   int     common_flag=0      // domaine d'action
   );
```

Paramètres

folder_name

[in] Le nom du répertoire,où il faut supprimer tous les fichiers. Contient le chemin complet vers le dossier.

common_flag=0

[in] Le drapeau, définissant la situation du répertoire. Si common_flag=FILE_COMMON, le répertoire se trouve dans le dossier commun de tous les terminaux de client \Terminal\Common \Files. Dans le cas contraire le répertoire se trouve dans le dossier local (MQL5\files ou MQL5 \tester\files en cas du test).

La valeur rendue

Rend true en cas du succès, autrement rend false.

Note

Se servez prudemment de cette fonction, puisque tous les fichiers et tous les sous-répertoires inclus se suppriment irrévocablement.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                                Demo_FolderClean.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- la description
#property description "Le script montre un exemple de l'utilisation FolderClean()."
#property description "D'abord on crée les fichiers dans le dossier indiqué à l'aide d
#property description "Puis, avant la suppression des fichiers on déduit l'avertisseme

//--- Nous montrons la fenêtre des paramètres d'entrée au lancement du script
#property script_show_inputs
//--- les paramètres d'entrée
input string   foldername="demo_folder";  // créons le dossier dans MQL5/Files/
input int      files=5;                   // Combien de fichiers nous créons et nous s
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string name="testfile";
//--- d'abord nous ouvrirons ou nous créons les fichiers dans le dossier des données d
   for(int N=0;N<files;N++)
     {
      //--- collectons le nom du fichier comme 'demo_folder\testfileN.txt'
      string filemane=StringFormat("%s\\%s%d.txt",foldername,name,N);
      //--- ouvrons le fichier avec le drapeau sur l'enregistrement, dans ce cas le do
      int handle=FileOpen(filemane,FILE_WRITE);
      //--- connaissons, autant la fonction a été accompli avec succès FileOpen()
      if(handle==INVALID_HANDLE)
        {
         PrintFormat("On n'a pas réussi à créer le fichier %s. Le code de l'erreur ",f
         ResetLastError();
        }
      else
        {
         PrintFormat("Le fichier %s est ouvert avec succès",filemane);
         //--- nous n'avons plus besoin du fichier ouvert, il faut le fermer obligatoi
         FileClose(handle);
        }
     }

//---vérifions combien de fichiers sont dans le dossier
   int k=FilesInFolder(foldername+"\\*.*",0);
   PrintFormat("Au total%s on a trouvé %d fichiers",foldername,k);

//--- déduisons la boîte de dialogue et demanderons à l'utilisateur
   int choice=MessageBox(StringFormat("Vous allez supprimer %s %d fichiers du dossier,
                         "La suppression des fichiers du dossier",
                         MB_YESNO|MB_ICONQUESTION); //  il y aura deux boutons - "Yes"
   ResetLastError();

//--- accomplirons les actions en fonction de la variante choisie
   if(choice==IDYES)
     {
      //--- commençons à supprimer
      PrintFormat("La tentative de supprimer tous les fichiers du dossier %s",folderna
      if(FolderClean(foldername,0))
```

© 2000-2015, MetaQuotes Software Corp.

```
         PrintFormat("Les fichiers sont supprimés avec succès, %s fichiers %d sont rés
                     foldername,
                     FilesInFolder(foldername+"\\*.*",0));
      else
         PrintFormat("On n'a pas réussi à supprimer les fichiers du dossier %s. Le coc
     }
   else
      PrintFormat("La suppression est annulée");
//---
  }
//+------------------------------------------------------------------+
//| rend le nombre de fichiers dans le dossier                       |
//+------------------------------------------------------------------+
int FilesInFolder(string path,int flag)
  {
   int count=0;
   long handle;
   string filename;
//---
   handle=FileFindFirst(path,filename,flag);
//--- Si quand même un fichier est trouvé, nous cherchons les autres
   if(handle!=INVALID_HANDLE)
     {
      //--- déduisons le nom du fichier
      PrintFormat("le fichier %s est trouvé",filename);
      //--- accroissons le compteur des fichiers/dossiers trouvés
      count++;
      //--- commençons le balayage de tous les fichiers/dossiers
      while(FileFindNext(handle,filename))
        {
         PrintFormat("le fichier %s a été trouvé",filename);
         count++;
        }
      //--- fermons obligatoirement le handle de la recherche à la fin
      FileFindClose(handle);
     }
   else // on n'a pas réussi à recevoir le handle
     {
      PrintFormat("On n'a pas réussi à faire la recherche des fichiers dans le dossier
     }
//--- rendons le résultat
   return count;
  }
```

Voir aussi

FileFindFirst, FileFindNext, FileFindClose

© 2000-2015, MetaQuotes Software Corp.

Les indicateurs d'utilisateur

Le groupe des fonctions utilisées lors de la présentation des indicateurs d'utilisateur. On ne peut pas utiliser ces fonctions pour écrire les conseillers et les scripts.

Fonction	Action
SetIndexBuffer	Attache le tampon d'indicateur indiquée avec le tableau unidimentionnel dynamique du type double
IndicatorSetDouble	Spécifie la valeur de la propriété d'indicateur, ayant le type double
IndicatorSetInteger	Spécifie la valeur de la propriété d'indicateur, ayant le type int
IndicatorSetString	Spécifie la valeur de la propriété d'indicateur, ayant le type string
PlotIndexSetDouble	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type double
PlotIndexSetInteger	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type int
PlotIndexSetString	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type string
PlotIndexGetInteger	Rend la valeur de la propriété de la ligne d'indicateur, ayant le type entier

Propriétés des indicateurs peuvent être installées à l'aide des directives de la compilation, et à l'aide des fonctions. Pour mieux comprendre il est recommandé d'étudierles styles des indicateurs dans les exemples.

Il est nécessaire de placer tous les comptes nécessaires des indicateurs d'utilisateur dans la fonction prédéterminée OnCalculate(). Si on utilise la forme courte de l'appel de la fonction OnCalculate() de l'aspect suivant

```
int OnCalculate (const int rates_total, const int prev_calculated, const int begin, co
```

la variable *rates_total* contient la valeur du nombre total d'éléments du tableau price[], transmis à titre du paramètre d'entrée pour le calcul des valeurs d'indicateur.

Le paramètre *prev_calculated* - c'est le résultat de l'exécution de la fonction OnCalculate() sur l'appel précédent et permet d'organiser l'algorithme économe du calcul des valeurs d'indicateur. Par exemple, si la valeur courante rates_total=1000, et prev_calculated=999, probablement, il nous suffit de faire les calculs seulement pour une valeur de chaque tampon d'indicateur.

Si l'information sur la grandeur du tableau d'entrée price était inaccessible, cela a amené à la nécessité de faire les calculs pour les 1000 valeurs de chaque tampon d'indicateur. Au premier appel de la fonction OnCalculate() la valeur prev_calculated=0. Si le tableau price[] a changé d'une façon ou d'une autre,dans ce cas prev_calculated est aussi égal au 0.

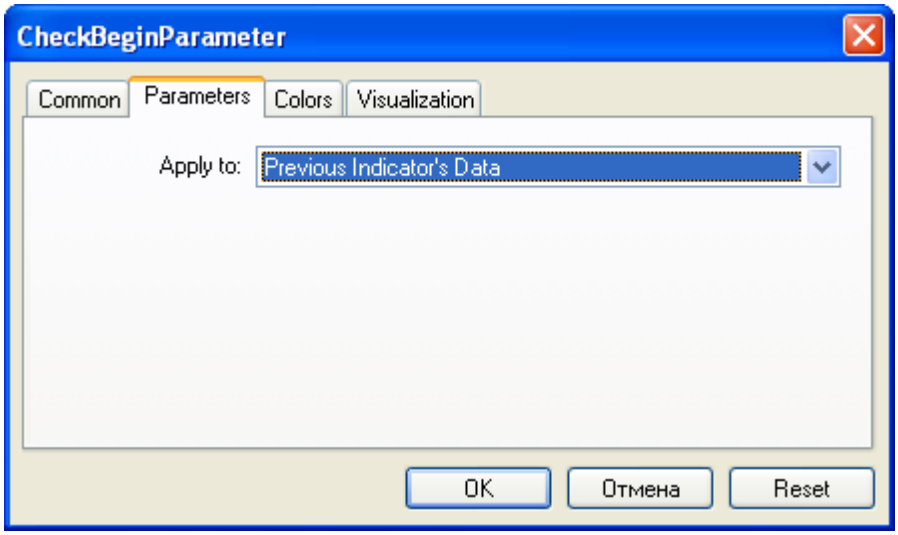
© 2000-2015, MetaQuotes Software Corp.

Le paramètre *begin* montre le nombre de valeurs initiales du tableau price, qui ne contient pas de données pour le calcul. Par exemple, si à titre du du tableau d'entrée on utilisait les valeurs de l'indicateur Accelerator Oscillator (pour lesquelles les premiers 37 valeurs ne sont pas calculées), begin=37. Comme l'exemple nous examinerons l'indicateur simple:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])

  {
//---
   Print("begin = ",begin,"  prev_calculated = ",prev_calculated,"  rates_total = ",ra
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Traînez-le de la fenêtre "Navigateur" à la fenêtre de l'indicateur Accelerator Oscillator et indiquons que les calculs seront produits sur les valeurs de l'indicateur précédent:

© 2000-2015, MetaQuotes Software Corp.



A la suite du premier appel de la fonction OnCalculate() la valeur prev_calculated sera égal au zéro, mais aux appels ultérieurs il sera égal à la valeur rates_total (jusqu'à ce que le nombre de barres sur le graphique de prix n'augmentera pas).

La valeur du paramètre begin sera en exactitude égal au nombre de barres initialles, pour lesauelles les valeurs de l'indicateur Accelerator ne sont pas calculées conformément à la logique de cet indicateur. Si nous regardons le code primaire de l'indicateur d'utilisateur Accelerator.mq5, nous verrons dans la fonction OnInit() les telles lignes:

```
//--- sets first bar from what index will be drawn
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,37);
```

Notamment par la fonction PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, empty_first_values)nous spécifions le nombre de premières valeurs irrelevantes au tableau nul d'indicateur de l'indicateur

© 2000-2015, MetaQuotes Software Corp.

d'utilisateur, lesquelles il ne faut pas prendre en attention pour les calculs (empty_first_values). Ainsi, nous avons des mécanismes pour:

1. définissez le nombre de valeurs de l'indicateur, qui ne devraient pas être utilisé pour les calculs dans un autre indicateur d'utilisateur;
2. recevoir l'information sur le nombre de premières valeurs, qu'il est nécessaire d'ignorer à l'appel de l'autre indicateur d'utilisateur, sans entrer dans la logique de ses calculs.

© 2000-2015, MetaQuotes Software Corp.

Styles des indicateurs dans les exemples

Dans le terminal de client MetaTrader 5 il y a 38 indicateurs techniques qui peuvent être utilisés dans les programmes MQL5 à l'aidedes fonctions correspondantes. Mais un principal avantage du langage MQL5 - c'est la possibilité de créer ses propres indicateurs d'utilisateur, qui peuvent ensuite être utilisées dans les conseillers pour recevoir des valeurs ou tout simplement d'imposer sur les graphiques de prix pour l'analyse technique.

On peut recevoir l'ensemble des indicateurs sur la base de quelquesstyles de la dessin, appelés appelé comme les constructions graphiques. La construction est considérée comme le moyen de l'affichage des données, lesquelles l'indicateur calcule, garde et donne selon la demande. Au total il y a sept constructions de base:

1. la ligne,
2. la section (le segment),
3. l'histogramme,
4. la flèche (le symbole)
5. le domaine peint (le canal avec le remplissage),
6. les barres,
7. les chandeliers japonnais.

Chaque construction exige pour son affichage de 1 à cinq tableauxdu type doubleoù se trouvent les valeurs de l'indicateur. Ces tableaux se connectent aux tampons d'indicateur pour le confort du fonctionnement de l'indicateur. Il faut déclarer le nombre de tampons dans l'indicateur à l'aide de ladirective du compilateur, par exemple:

```
#property indicator_buffers 3 // le nombre de tampons
#property indicator_plots   2 // le nombre de constructions graphiques
```

Le nombre de tampons dans l'indicateur est toujours plus ou égal au nombre de constructions dans l'indicateur.

Comme chaque construction de base graphique peut avoir les variations de couleur ou l'affichage spécifique, le nombre réel de constructions dans le langage de MQL5 est 18:

La construction	La description	Les tampons des valeurs	Des tampons de la couleur
DRAW_NONE	N'affiche pas visuellement sur le graphique, mais on peut regarder les valeurs du tampon correspondant dans "Fenêtre des données"	1	-
DRAW_LINE	On construit la ligne selon les valeurs du tampon correspondant (les valeurs vides dans le tampon sont	1	-

© 2000-2015, MetaQuotes Software Corp.

	indésirables)		
DRAW_SECTION	Se dessine par les segments entre les valeurs du tampon correspondant (d'habitude il y a beaucoup de valeurs vides)	1	-
DRAW_HISTOGRAM	Se dessine par l'histogramme de la ligne zéro aux valeurs du tampon correspondant (on admet les valeurs vides)	1	-
DRAW_HISTOGRAM2	Se dessine par l'histogramme sur deux tampons d'indicateur (on admet les valeurs vides)	2	-
DRAW_ARROW	S'affiche par les symboles (on admet les valeurs vides)	1	-
DRAW_ZIGZAG	Est semblable au style DRAW_SECTION, mais à la différence de lui il peut construire les segments verticaux sur une barre	2	-
DRAW_FILLING	Remplissage de couleur entre les deux lignes. Dans "La fenêtre des données" sont affichées les valeurs des deux tampons correspondants	2	-
DRAW_BARS	L'affichage en forme de barres sur le graphique. Dans "La fenêtre des données" sont montrées les 4 valeurs des tampons correspondants	4	-
DRAW_CANDLES	L'affichage en forme	4	-

© 2000-2015, MetaQuotes Software Corp.

	de chandeliers japonais. Dans "La fenêtre des données" sont montrées les 4 valeurs des tampons correspondants		
DRAW_COLOR_LINE	La ligne pour laquelle vous pouvez alterner les couleurs sur les barres différentes ainsi que changer sa couleur à n'importe quel moment du temps	1	1
DRAW_COLOR_SECTION	Est semblable au styleDRAW_SECTION, mais on peut spécifier la couleur de chaque section individuellement, on peut spécifier la couleur dynamiquement	1	1
DRAW_COLOR_HISTOGRAM	Est semblable au styleDRAW_HISTOGRAM, mais chaque bande peut avoir la couleur, on peut spécifier la couleur dynamiquement	1	1
DRAW_COLOR_HISTOGRAM2	Est semblable au style DRAW_HISTOGRAM2, mais chaque bande peut avoir la couleur, on peut spécifier la couleur dynamiquement	2	1
DRAW_COLOR_ARROW	Est semblable au styleDRAW_ARROW, mais chaque symbole peut avoir sa couleur. La couleur peut être changée dynamiquement	1	1
DRAW_COLOR_ZIGZAG	Le styleDRAW_ZIGZAG avec la possibilité du	2	1

© 2000-2015, MetaQuotes Software Corp.

	coloriage individuel des sections et le remplacement dynamique de la couleur		
DRAW_COLOR_BARS	Le styleDRAW_BARSavec la possibilité du coloriage individuel des barres et le remplacement dynamique de la couleur	4	1
DRAW_COLOR_CANDLES	Le styleDRAW_CANDLES avec la possibilité du coloriage individuel des bougies et le remplacement dynamique de la couleur	4	1

La différence entre le tampon d'indicateur et le tableau

Il faut déclarer dans chaque indicateur auniveau global un ou plusieurs tableaux de type double, qui doit être utilisé après comme un tampon d'indicateur à l'aide de la fonctionSetIndexBuffer(). Pour dessiner les constructions graphiques de l'indicateur sont utilisés seulement les valeurs de tampons d'indicateur, ou on ne peut pas utiliser les autres tableaux pour cela. En outre, les valeurs des tampons sont montrées dans "la Fenêtre des données".

Un tampon d'indicateur doit être dynamique et ne nécessite pas l'indication de la taille - la taille du tableau utilisé comme le tampon d'indicateur, s'installe automatiquement par le sous-système exécutif du terminal.

La direction de l'indexation après le binding du tableau avec le tampon d'indicateur s'établit par défaut comme dans les tableaux ordinaires, mais on peut appliquer en cas de nécessité la fonctionArraySetAsSeries() pour le changement du moyen de l'accès aux éléments du tableau. Par défaut, le tampon d'indicateur est utilisé pour stocker les données destinés pour le dessin (INDICATOR_DATA .)

Si pour les calculs des valeurs de l'indicateur il faut faire les calculs intermédiaires et garder la valeur auxiliaire pour chaque barre, pendant le binding on peut déclarer un tel tableau à titre du tampon de calcul (INDICATOR_CALCULATIONS). On peut utiliser un tableau ordinaire pour des valeurs intermédiaires, mais dans ce cas le programmeur doit gérer la taille de tel tableau.

Certaines constructions permettent de spécifier pour chaque barre la couleur de l'image. Pour stocker l'information sur la couleur on utilise les tampons de couleur (INDICATOR_COLOR_INDEX_.) La couleur est représenté par un type entier color, mais tous les tampons d'indicateurs doivent avoir le typedouble. On ne peut pas recevoir les valeurs colorées et auxiliaires (INDICATOR_CALCULATIONS)

© 2000-2015, MetaQuotes Software Corp.

des tampons à l'aide de la fonctionCopyBuffer().

Le nombre de tampons d'indicateur doit être indiqué par la directive du compilateur#property indicator_buffers le nombre_de tampons:

```
#property indicator_buffers 3  //  l'indicateur a 3 tampons
```

Le nombre maximum de tampons dans un indicateur est 512.

La conformité des tampons d'indicateur et des constructions graphiques

Chaque construction graphique est basé sur un ou plusieurs tampons d'indicateur. Ainsi pour un simple affichage des chandeliers japonais il est nécessaire quatre valeurs - les prix Open, High, Low et Close. En conséquence, pour afficher l'indicateur en forme de chandeliers japonais il faut déclarer 4 tampons d'indicateur, et 4 tableaux de type double pour eux. Par exemple:

```
//--- il y a quatre tampons d'indicateur dans l'indicateur
#property indicator_buffers 4
//--- il y a une construction graphique dans un indicateur
#property indicator_plots   1
//--- la construction graphique sous le numéro 1 s'affichera par les chandeliers japor
#property indicator_type1   DRAW_CANDLES
//--- Les chandeliers japonaises se dessineront par la couleur clrDodgerBlue
#property indicator_color1  clrDodgerBlue
//--- 4 tableaux pour les tampons d'indicateur
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

Les constructions graphiques utilisent automatiquement les tampons d'indicateur conformément au numéro de la construction. Les numéros de la construction commencent par 1, les numéros des tampons commencent par le zéro. Si la première construction nécessite 4 tampons d'indicateur, les 4 premiers tampons d'indicateur seront utilisés pour le dessin. Ces quatre tampons doivent être liés par la fonction SetIndexBuffer()aux tableaux correspondants avec l'indexation juste.

```
//--- l'enchaînement des tableaux avec les tampons d'indicateur
   SetIndexBuffer(0,OBuffer,INDICATOR_DATA);  // le premier tampon correspond à l'inde
   SetIndexBuffer(1,HBuffer,INDICATOR_DATA);  // le deuxième tampon correspond à l'ind
   SetIndexBuffer(2,LBuffer,INDICATOR_DATA);  // le troisième tampon correspond à l'in
   SetIndexBuffer(3,CBuffer,INDICATOR_DATA);  // le quatrième tampon correspond à l'in
```

Pendant qu'on fait le dessin des chandeliers japonais l'indicateur utilisera notamment les premiers quatre tampons, parce que la construction "les chandeliers japonais" a été déclarée par le premier numéro.

Changeons un peu l'exemple, ajoutons la construction en forme de la ligne simple - DRAW_LINE. Supposons maintenant que la ligne aura le numéro 1 et les chandeliers japonais seront sous le numéro

© 2000-2015, MetaQuotes Software Corp.

2. Le nombre de tampons et le nombre de constructions a augmenté.

```
//--- dans l'indicateur il y a 5 tampons d'indicateur
#property indicator_buffers 5
//--- dans l'indicateur il y a 2 constructions graphiques
#property indicator_plots   2
//--- la construction graphique sous le numéro 1 s'affichera par la ligne
#property indicator_type1   DRAW_LINE
//--- la ligne sera dessinée par la couleur clrDodgerRed
#property indicator_color1  clrDodgerRed
//--- la construction graphique sous le numéro 2 s'affichera par les chandeliers japor
#property indicator_type2   DRAW_CANDLES
//--- Les chandeliers japonaises se dessineront par la couleur clrDodgerBlue
#property indicator_color2  clrDodgerBlue
//--- 5 tableaux pour les tampons d'indicateur
double LineBuffer[];
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

L'héritage des constructions a changé, maintenant par le premier il y a une ligne, après suivent les chandeliers japonais. C'est pourquoi l'ordre de l'héritage des tampons sera le même - d'abord nous déclarons le tampon pour la ligne sous l'index zéro, et puis quatre tampons pour l'affichage des chandeliers japonais.

```
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);  // le deuxième tampon correspond à l'
//---l'enchaînement des tableaux avec les tampons d'indicateur pour les chandeliers ja
   SetIndexBuffer(1,OBuffer,INDICATOR_DATA);     // le deuxième tampon correspond à l'
   SetIndexBuffer(2,HBuffer,INDICATOR_DATA);     // le troisième tampon correspond à l
   SetIndexBuffer(3,LBuffer,INDICATOR_DATA);     // le quatrième tampon correspond à l
   SetIndexBuffer(4,CBuffer,INDICATOR_DATA);     // le cinquième tampon correspond à l
```

On peut spécifier le nombre de tampons et de constructions graphiques seulement à l'aide des directives du compilateur, le changement dynamique de ces propriétés à l'aide des fonctions est impossible.

Les versions colorées des styles

Comme on peut voir dans le tableau, les styles se divisent sur deux groupes. Le premier groupe — ce sont les styles dont dans ses titres il n'y a pas le mot **COLOR**, appellerons-les les styles de base:

- DRAW_LINE
- DRAW_SECTION
- DRAW_HISTOGRAM
- DRAW_HISTOGRAM2
- DRAW_ARROW
- DRAW_ZIGZAG

© 2000-2015, MetaQuotes Software Corp.

- DRAW_FILLING
- DRAW_BARS
- DRAW_CANDLES

Le deuxième groupe des styles contient dans le nom le mot **COLOR**, appellerons-les leurs versions colorées:

- DRAW_COLOR_LINE
- DRAW_COLOR_SECTION
- DRAW_COLOR_HISTOGRAM
- DRAW_COLOR_HISTOGRAM2
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG
- DRAW_COLOR_BARS
- DRAW_COLOR_CANDLES

Toutes les versions colorées des styles se distinguent des sujets de base qui permettent de spécifier la couleur pour chaque partie de la construction graphique. La partie minimale de la construction est le barre, c'est pourquoi on peut dire que les versions colorées permettent de spécifier la couleur de la construction sur chaque barre.

Pour spécifier la couleur de la construction sur chaque barre, aux versions colorées des styles a été ajouté le tampon supplémentaire spécial pour stocker l'index de la couleur. Ces index indiquent au numéro de la couleur dans un tableau spécial contenant l'ensemble prédéfini de couleurs. La taille du tableau des couleurs est 64. Cela signifie que chaque version colorée du style permet de colorer la construction graphique par 64 couleurs différentes.

On peut spécifier l'ensemble et le nombre de couleurs dans un tableau spécial des couleurs par la directive du compilateur #property indicator_color, où on indique toutes les couleurs nécessaires par la virgule. Par exemple, une telle inscription dans l'indicateur:

```
//--- spécifions 8 couleurs pour colorer les bougies (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
```

Ici est indiqué que pour la construction graphique le numéro 1 on spécifie 8 couleurs, qui seront placées au tableau spécial. Ensuite dans le programme nous spécifions pas la couleur elle-même, par laquelle s'affichera la construction graphique, mais seulement son index. Si nous voulons spécifierla couleur rouge pour la barre, pour cela il est nécessaire d'installer l'index de la couleur rouge du tableau dans un tampon coloré. La couleur rouge est spécifiée dans la directive par le première, l'index le numéro 0 lui correspond.

```
  //--- spécifions la couleur de la bougie clrRed
  col_buffer[buffer_index]=0;
```

L'ensemble des couleurs pour le coloriage n'est pas spécifié une fois pour toujours, on peut le changer dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Exemple:

```
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
```

© 2000-2015, MetaQuotes Software Corp.

```
                                color_array[i]);      //  une nouvelle couleur
```

Les propriétés de l'indicateur et des constructions graphiques

Pour les constructions graphiques on peut installer les propriétés à l'aidedirectives du compilateur, ainsi qu'à l'aide des fonctions appropriées. Plus en détail cela est décrit dans le paragrapheLien entre les propriétés de l'indicateur et les fonctions. Le changement dynamique des propriétés de l'indicateur à l'aide des fonctions permet de créer les indicateurs plus flexibles d'utilisateur.

Commencer à dessiner l'indicateur sur le graphique

En plusieurs cas sous les conditions de l'algorithme il est impossible de commencer le calcul des valeurs de l'indicateur tout de suite par la barre actuelle, il faut assurer le minimum des bars précédents accessibles dans l'histoire. Par exemple, plusieurs aspects de lissage sous-entendent qu'on prend le tableau des prix aux barres N précédentes, et sur la base de ces valeurs est calculée la valeur de l'indicateur pour une barre actuelle.

Dans tels cas il n'y a pas de possibilité de calculer les valeurs de l'indicateur sur les premières barres N, ou ces valeurs ne sont pas destinées à l'affichage sur le graphique et ils sont seulement auxiliaire pour le calcul des valeurs ultérieures. Pour refuser la visualisation de l'indicateur sur les premières barres N de l'histoire, il faut mettre à la propriété PLOT_DRAW_BEGIN la valeur N pour la construction correspondante graphique:

```
//---l'enchaînement des tableaux avec les tampons d'indicateur pour les chandeliers ja
PlotIndexSetInteger(le numéro_de la construction_graphique,PLOT_DRAW_BEGIN,N);
```

Ici:

- le numéro_de la construction_graphique – la valeur du zéro jusqu'au indicator_plots-1 (le numérotage des constructions graphiques commence par le zéro).
- N – le nombre de premières barres dans l'histoire, sur lesquelles l'indicateur ne doit pas s'afficher sur le graphique.

© 2000-2015, MetaQuotes Software Corp.

DRAW_NONE

Le style DRAW_NONE est conçu pour les cas quand il est nécessaire de calculer et afficher les valeurs du tampon dans "La fenêtre des données", mais l'affichage sur le graphique n'est pas nécessaire. Pour le réglage de l'exactitude de l'affichage utilisez l'expression IndicatorSetInteger(INDICATOR_DIGITS,le nombre_des signes)dans la fonction OnInit() :

```
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- établissons l'exactitude, avec laquelle la valeur sera affichée dans la Fenêtre
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
```

Le nombre de tampons nécessaires pour la construction DRAW_NONE — 1.

L'exemple de l'indicateur, qui affiche dans "la Fenêtre des données" le numéro de la barre, sur lequelle se trouve la souris. La numérotation correspond à la série temporelle, c'est-à-dire la barre inachevée actuel a l'index null, et la plus ancienne barre a le plus grand index.

Prêtez attention que malgré le fait qu'à la construction graphique №1 est spécifiée la couleur rouge de l'affichage, l'indicateur ne dessine rien sur le graphique.

```
//+------------------------------------------------------------------+
//|                                                    DRAW_NONE.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Invisible
#property indicator_label1  "Bar Index"
#property indicator_type1   DRAW_NONE
#property indicator_style1  STYLE_SOLID
#property indicator_color1  clrRed
#property indicator_width1  1
//--- indicator buffers
double         InvisibleBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le binding du tableau et du tampon d'indicateur
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- établissons l'exactitude, avec laquelle la valeur sera affichée dans "la Fenêtre
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static datetime lastbar=0;
//--- si c'est le premier calcul de l'indicateur
   if(prev_calculated==0)
     {
      //--- numéroterons les barres pour la première fois
      CalcValues(rates_total,close);
      //--- retiendrons le temps de l'ouverture de la barre actuelle dans lastbar
      lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
    else
      {
       //---s'il y avait une nouvelle barre, le temps de son ouverture ne coïncide pas
       if(lastbar!=SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE))
         {
          //--- numéroterons les barres de nouveau
          CalcValues(rates_total,close);
          //---mettons à jour le temps de l'ouverture de la barre actuelle dans lastbar
          lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
         }
      }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| numérote les barres comme dans la série temporelle
//+------------------------------------------------------------------+
void CalcValues(int total,double const  &array[])
  {
//--- spécifions l'indexation au tampon d'indicateur comme dans la série temporelle
   ArraySetAsSeries(InvisibleBuffer,true);
//--- remplirons à chaque barre son numéro
   for(int i=0;i<total;i++) InvisibleBuffer[i]=i;
  }
```

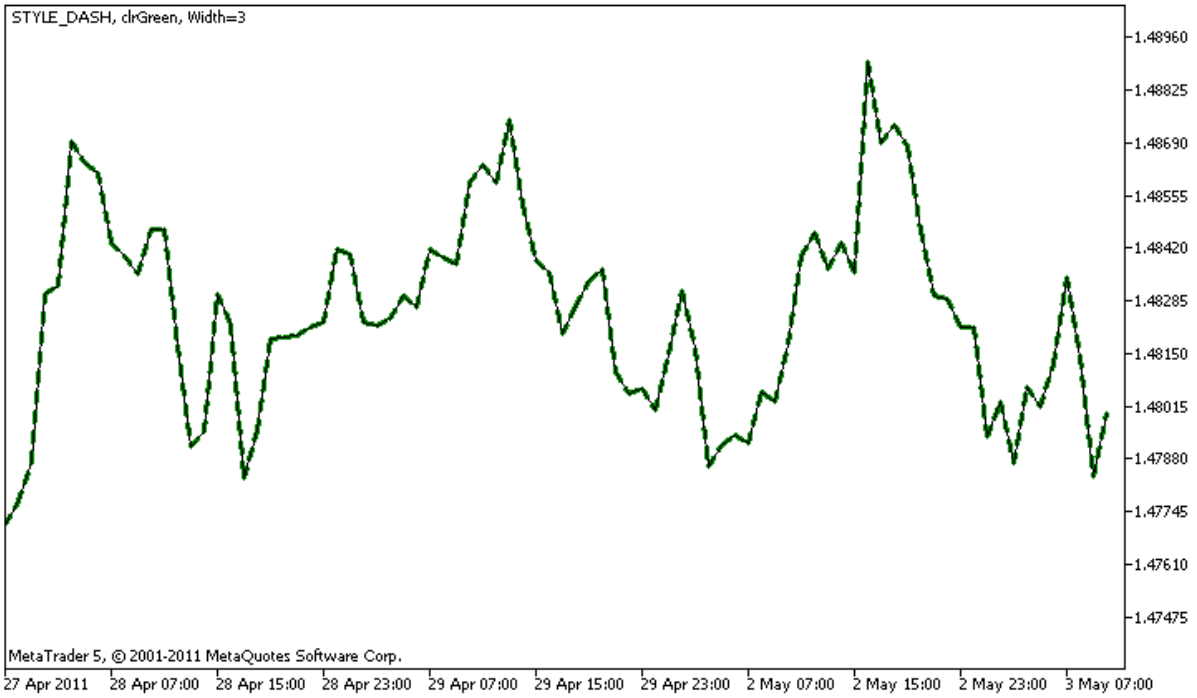
© 2000-2015, MetaQuotes Software Corp.

DRAW_LINE

Le style DRAW_LINE dessine par la couleur spécifiée la ligne selon les valeurs du tampon d'indicateur. On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la ligne comme par les directives du compilateur, et dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

Le nombre de tampons nécessaires pour la construction DRAW_LINE — 1.

L'exemple de l'indicateur dessinant la ligne aux prix de la clôture des barres Close. La couleur, l'épaisseur et le style de la ligne changent chaques 5 ticks par hasard.



Prêtez attention que primordialement à la construction graphique**plot1** avec le style DRAW_LINE on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() ces trois propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                                    DRAW_LINE.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_LINE"
#property description "Dessine la ligne par la couleur spécifiée selon les prix Close'
#property description "La couleur, l'épaisseur et le style de la ligne change par hasa
```

© 2000-2015, MetaQuotes Software Corp.

```
#property description "dans chaques N ticks"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- les propriétés de la ligne sont spécifiées à l'aide des directives du compilateu
#property indicator_label1  "Line"      // le nom de la construction pour "la Fenêtre
#property indicator_type1   DRAW_LINE   // le type de la construction graphique - la l
#property indicator_color1  clrRed      // la couleur de la ligne
#property indicator_style1  STYLE_SOLID // le style des lignes
#property indicator_width1  1           // l'épaisseur de la ligne
//--- le paramètre input
input int      N=5;          // le nombre de ticks pour le changement
//--- le tampon d'indicateur pour la construction
double         LineBuffer[];
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le binding du tableau et du tampon d'indicateur
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//--- l'initialisation du générateur des nombres pseudo-accidentels
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseu
   ticks++;
//--- si le nombre critique des ticks est accumulé
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le bloc du calcul des valeurs de l'indicateur
   for(int i=0;i<rates_total;i++)
     {
      LineBuffer[i]=close[i];
     }

//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| modifie l'apparence de la ligne affichée dans l'indicateur       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de la couleur de la ligne
//--- recevrons le nombre accidentel
   int number=MathRand();
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la ligne
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+", Width="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   size=ArraySize(styles);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm=EnumToString(styles[style_index])+", "+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_SECTION

Le style DRAW_SECTION dessine par la couleur spécifiée les segments selon les valeurs du tampon d'indicateur. On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la ligne comme pour le style DRAW_LINE - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

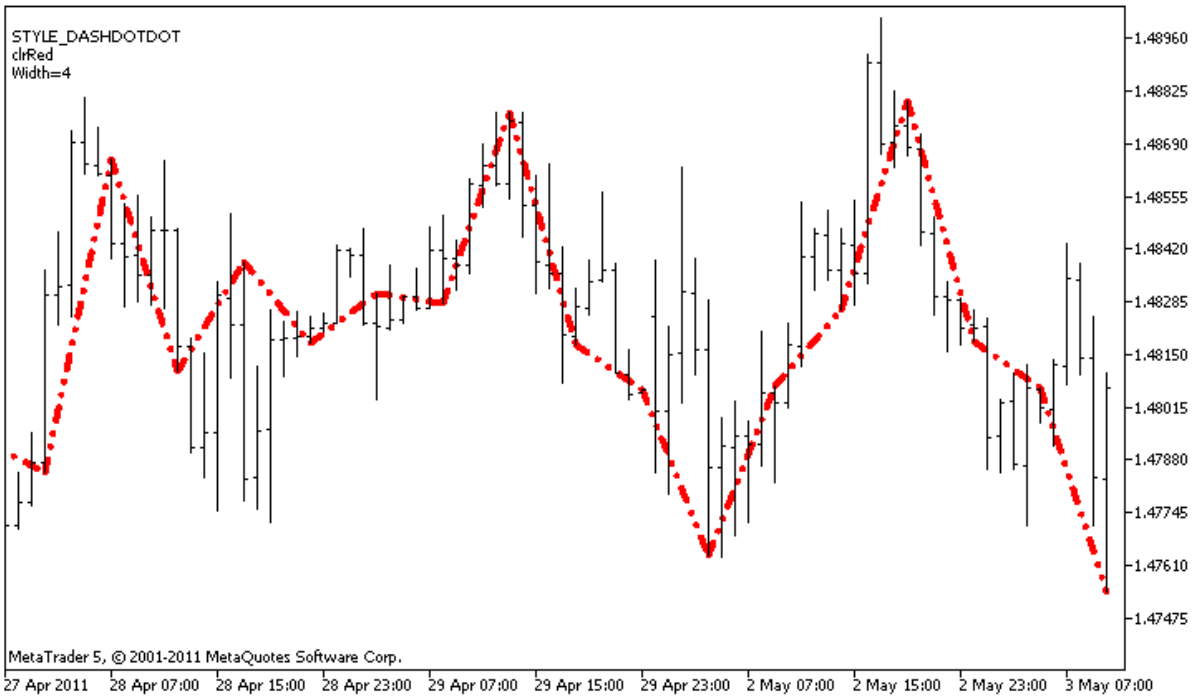
Les sections se dessinent d'une valeur non vide à l'autre valeur non vide du tampon d'indicateur, les valeurs vides sont manquées. Pour indiquer, quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriété PLOT_EMPTY_VALUE. Par exemple, si l'indicateur doit se dessiner par les segments selon les valeurs non nulles, il faut spécifier la valeur nulle comme vide:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(l'index_de la construction_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs tous les éléments du tampon d'indicateur, spécifiez aux éléments non-rendus la valeur vide.

Le nombre de tampons nécessaires pour la construction DRAW_SECTION — 1.

L'exemple de l'indicateur dessinant les segments entre les prix High et Low. La couleur, l'épaisseur et le style de **toutes** les sections se changent chaques**N** ticks par hasard.



Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_SECTION on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() ces trois propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                                DRAW_SECTION.mq5 |
```

© 2000-2015, MetaQuotes Software Corp.

```
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_SECTION"
#property description "Dessine par les segments directs dans chaque "bars" barres"
#property description "La couleur, l'épaisseur et le style des sections se changent pa
#property description "dans chaques N ticks"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Section
#property indicator_label1  "Section"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- le paramètre input
input int      bars=5;           // la longueur des sections dans les barres
input int      N=5;              // le nombre de ticks pour le changement du style des
//--- le tampon d'indicateur pour la construction
double         SectionBuffer[];
//---la variable auxiliaire pour le calcul des fins des sections
int            divider;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le binding du tableau et du tampon d'indicateur
   SetIndexBuffer(0,SectionBuffer,INDICATOR_DATA);
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---vérifions le paramètre de l'indicateur
   if(bars<=0)
     {
      PrintFormat("La valeur inadmissible du paramètre bar=%d",bars);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=2*bars;
//---+
```

© 2000-2015, MetaQuotes Software Corp.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le numéro de la barre, par laquelle nous commencerons le calcul des valeurs de l
   int start=0;
//--- si l'indicateur a été déjà calculé, établissons start sur la barre précédente
   if(prev_calculated>0) start=prev_calculated-1;
//--- Tous les calculs des valeurs de l'indicateur se trouvent ici
   for(int i=start;i<rates_total;i++)
     {
      //--- recevrons le reste de la division du numéro de la barre sur 2*bars
      int rest=i%divider;
      //--- si le numéro de la barre se divise sans reste sur 2*bars
      if(rest==0)
        {
         //---fixerons la fin du segment sur le prix High de cette barre
         SectionBuffer[i]=high[i];
        }
      //--- si le reste de la division est égal aux barres,
      else
        {
         //---fixerons la fin du segment sur le prix High de cette barre
         if(rest==bars) SectionBuffer[i]=low[i];
         //--- si rien n'est pas convenu, manquons cette barre - mettons la valeur 0
```

© 2000-2015, MetaQuotes Software Corp.

```
         else SectionBuffer[i]=0;
        }
      }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| modifie l'apparence des sections dans l'indicateur               |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de la couleur de la ligne
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la ligne
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- établissons l'épaisseur
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- établissons le style des lignes
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

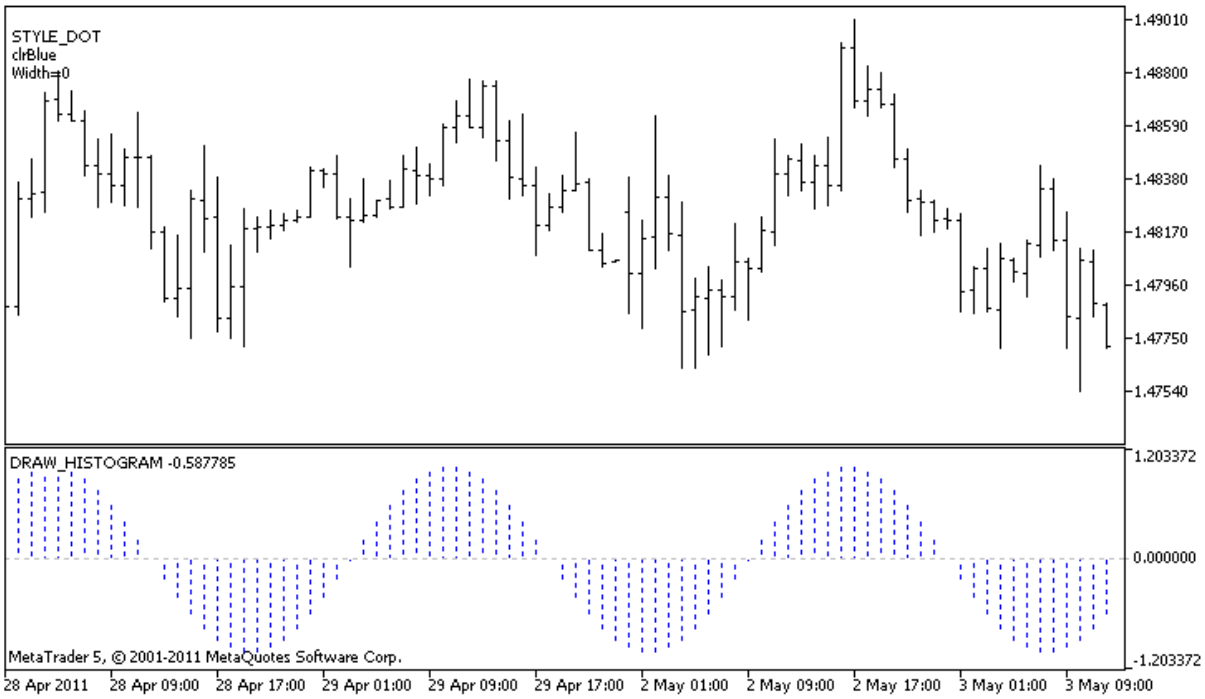
DRAW_HISTOGRAM

Le style DRAW_HISTOGRAM dessine par la couleur spécifiée l'histogramme par les colonnes du zéro jusqu'à la valeur indiquée. Les valeurs sont tirées du tampon de l'indicateur. On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la colonne comme pour le style DRAW_LINE - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'histogramme en fonction de la situation actuelle.

Puisque sur chaque barre se dessine la colonne du niveau nul, alors il est mieux utiliser DRAW_HISTOGRAM pour l'affichage dans une sous- fenêtre séparée du graphique. Le plus souvent ce type de la construction graphique est utilisé pour la création des indicateurs du type d'oscillateur, par exemple, Bears Power ou OsMA. Il suffit de spécifier la valeur nulle pour les valeurs vides non affichées.

Le nombre de tampons nécessaires pour la construction DRAW_HISTOGRAM — 1.

L'exemple de l'indicateur, dessinant la sinusoïde selon la fonctionMathSin()par la couleur spécifiée. La couleur, l'épaisseur et le style de**toutes** colonnes de l'histogramme se changent chaques **N** ticks par hasard. Le paramètre "bars" spécifie la période de la sinusoïde, c'est-à-dire dans le nombre spécifié de barres la sinusoïde se répétera selon la boucle.



Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_HISTOGRAM on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonctionOnCalculate() ces trois propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
```

© 2000-2015, MetaQuotes Software Corp.

```
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_HISTOGRAM"
#property description "Dessine la sinusoïde par l'histogramme dans la fenêtre séparée"
#property description "La couleur, l'épaisseur des colonnes se changent par hasard"
#property description "dans chaques N ticks"
#property description "Le paramètre "bars" spécifie le nombre de barres dans la boucle

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres input
input int      bars=30;          // la période de la sinusoïde dans les barres
input int      N=5;              // le nombre de ticks pour le changement de l'histogr
//--- indicator buffers
double         HistogramBuffer[];
//--- le multiplicateur pour la réception de l'angle 2Pi dans les radians à la multipl
double    multiplier;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- calculerons le multiplicateur
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Spécifiez la valeur "bars"=%d plus que 1",bars);
      //--- une cessation anticipée du travail de l'indicateur
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- les calculs des valeurs de l'indicateur
   int start=0;
//--- si le calcul est déjà fait pendant le lancement précédent de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // établissons le début des calculs
//--- remplissons le tampon d'indicateur par les valeurs
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=sin(i*multiplier);
     }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| modifie l'apparence des lignes dans l'indicateur                 |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de la couleur de la ligne
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
```

© 2000-2015, MetaQuotes Software Corp.

```
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la ligne
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- établissons l'épaisseur
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- établissons le style des lignes
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_HISTOGRAM2

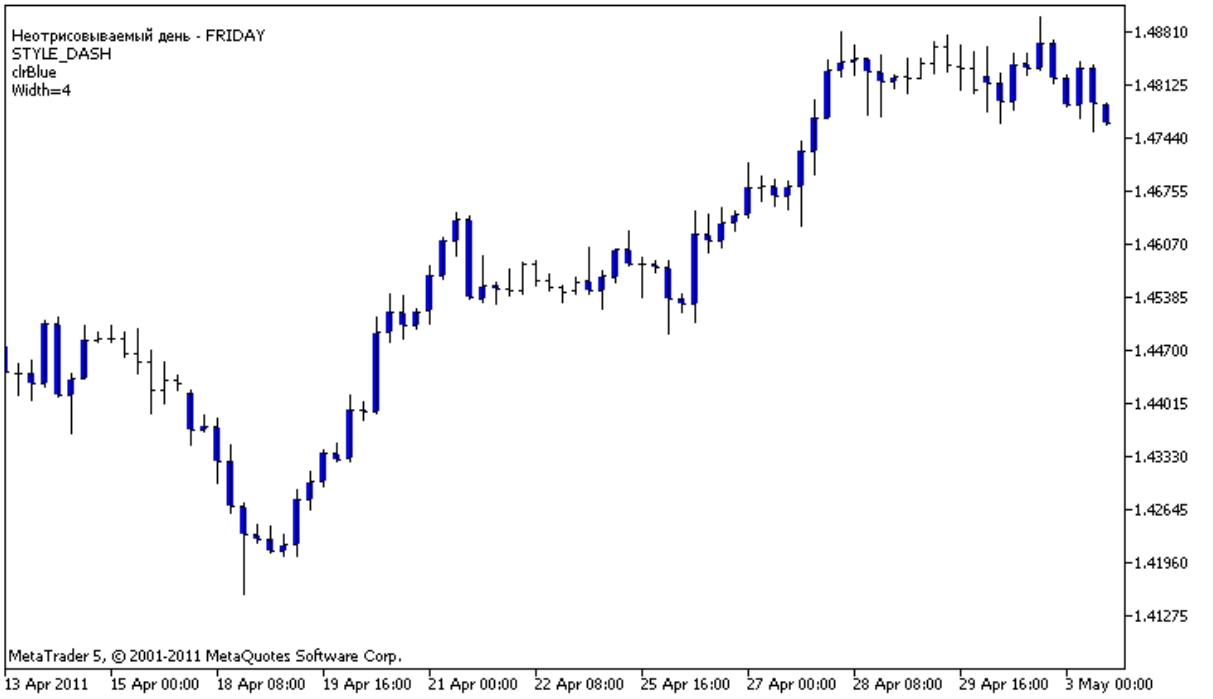
Le style DRAW_HISTOGRAM2 dessine l'histogramme par la couleur spécifiée - les segments verticaux selon les valeurs de deux tampons d'indicateur. On peut spécifier l'épaisseur, la couleur et le style de l'affichage des segments comme pour le style DRAW_LINE - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'histogramme en fonction de la situation actuelle.

On peut utiliser le style DRAW_HISTOGRAM dans une sous- fenêtre séparée du graphique, ainsi que dans une fenêtre principale. On ne fait pas le dessin pour les valeurs vides, toutes les valeurs dans les tampons d'indicateur doivent être installés de manière explicite. L'initialisation des tampons par la valeur vide n'est pas produite.

Le nombre de tampons nécessaires pour la construction DRAW_HISTOGRAM2 — 2.

L'exemple de l'indicateur dessinant sur chaque barre le segment vertical de la couleur spécifiée et les épaisseurs entre les prix Open et Close. La couleur, l'épaisseur et le style de **toutes** colonnes de l'histogramme se changent chaques **N** ticks par hasard. Au lancement de l'indicateur dans la fonction OnInit()le numéro du jour de la semaine s'établit par l'image accidentelle, auquel l'histogramme ne sera pas dessiné - invisible_day. La valeur vide PLOT_EMPTY_VALUE=0: est spécifiée pour cela

```
//---établissons la valeur vide
   PlotIndexSetDouble(l'index_de la construction_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```



Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_HISTOGRAM2 on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() ces trois propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//|                                                  DRAW_HISTOGRAM2.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                  https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_HISTOGRAM2"
#property description "Dessine le segment entre Open et Close sur chaque barre"
#property description "La couleur, l'épaisseur et le style se changent par hasard"
#property description "dans chaques N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot Histogram_2
#property indicator_label1  "Histogram_2"
#property indicator_type1   DRAW_HISTOGRAM2
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      N=5;              // le nombre de ticks pour le changement de l'histogr
//--- indicator buffers
double         Histogram_2Buffer1[];
double         Histogram_2Buffer2[];
//--- le jour de la semaine, pour lequel l'indicateur n'est pas dessiné
int invisible_day;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Histogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Histogram_2Buffer2,INDICATOR_DATA);
//---établissons la valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- recevrons le nombre accidentel de 0 jusqu'à 5
   invisible_day=MathRand()%6;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- les calculs des valeurs de l'indicateur
   int start=0;
//--- pour la réception du jour de la semaine selon le temps de l'ouverture de chaque
   MqlDateTime dt;
//--- si le calcul est déjà fait pendant le lancement précédent de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // établissons le début des calculs
//--- remplissons le tampon d'indicateur par les valeurs
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
      if(dt.day_of_week==invisible_day)
        {
         Histogram_2Buffer1[i]=0;
         Histogram_2Buffer2[i]=0;
        }
      else
        {
         Histogram_2Buffer1[i]=open[i];
         Histogram_2Buffer2[i]=close[i];
        }
     }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| modifie l'apparence des lignes dans l'indicateur                 |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de la couleur de la ligne
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la ligne
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- établissons l'épaisseur des lignes
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- établissons le style des lignes
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//---ajoutons l'information sur le jour, qui est manqué dans les calculs
   comm="\r\nLe jour non-dessiné - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+cor
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_ARROW

Le style DRAW_ARROW dessine les flèches sur le graphique par la couleur spécifiée (les symboles de l'ensemble Wingdings) selon la valeur du tampon d'indicateur. On peut spécifier l'épaisseur et la couleur des symboles de la meme façon que pour le style DRAW_LINE - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'indicateur en fonction de la situation actuelle.

Le code du symbole pour la sortie sur le graphique est spécifié à l'aide de la propriété PLOT_ARROW.

```
//--- - spécifions le code du symbole de l'ensemble Wingdings pour le dessin dans PLOT
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

Par défaut la valeur PLOT_ARROW=159 (le cercle).

Chaque flèche représente en réalité le symbole, qui a la hauteur et le point du rattachement, et peut fermer par lui-même une certaine importante information sur le graphique (par exemple, le prix de la clôture de la barre). C'est pourquoi on peut spécifier en supplément un décalage vertical en pixels, qui ne dépend pas de l'échelle du graphique. Les flèches seront visuellement déplacées selon la verticale sur le nombre indiqué de pixels, bien que les valeurs de l'indicateur restent les mêmes:

```
//--- spécifions le décalage des flèches selon le vertical en pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

La valeur négative PLOT_ARROW_SHIFT signifie le décalage des flèches en haut, la valeur positive déplace les flèches en bas.

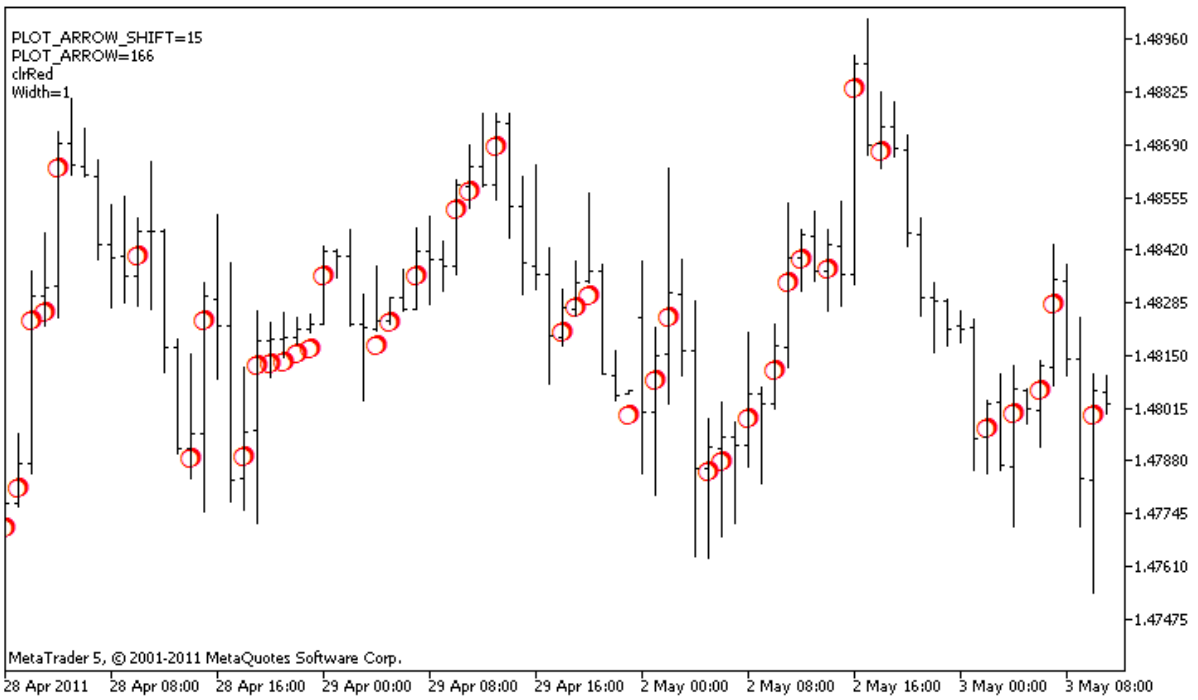
On peut utiliser le style DRAW_ARROW dans une sous- fenêtre séparée du graphique, ainsi que dans une fenêtre principale. Les valeurs vides ne se dessinent pas et ne s'affichent pas dans "la Fenêtre des données", toutes les valeurs dans les tampons d'indicateur doivent être installés de manière explicite. L'initialisation des tampons par la valeur vide n'est pas produite.

```
//---établissons la valeur vide
   PlotIndexSetDouble(l'index_de la construction_DRAW_ARROW,PLOT_EMPTY_VALUE,0);
```

Le nombre de tampons nécessaires pour la construction DRAW_ARROW — 1.

L'exemple de l'indicateur dessinant les flèches sur chaque barre, qui a le prix de la clôture Close plus grand que le prix de la clôture de la barre précédente La couleur, l'épaisseur et le code du symbole de**toutes**les flèches se changent chaques **N** ticks par hasard.

© 2000-2015, MetaQuotes Software Corp.



Dans l'exemple les propriétés la couleur et la taille pour la construction graphique **plot1** avec le style DRAW_ARROW sont spécifiés primordialement à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate()les propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_ARROW.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_ARROW"
#property description "Dessine les flèches, spécifiées par les symboles Unicode, sur l
#property description "La couleur, la taille, le décalage et le code du symbole de la
#property description "dans chaques N ticks"
#property description "Le paramètre code spécifie la valeur de base: le code = 159 (ce

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Arrows
#property indicator_label1  "Arrows"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrGreen
#property indicator_width1  1
//--- les paramètres input
```

© 2000-2015, MetaQuotes Software Corp.

```
input int      N=5;          // le nombre de ticks pour le changement
input ushort   code=159;     // le code du symbole pour le dessin dans DRAW_ARROW
//--- le tampon d'indicateur pour la construction
double         ArrowsBuffer[];
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ArrowsBuffer,INDICATOR_DATA);
//--- spécifions le code du symbole pour le dessin dans PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- spécifions le décalage des flèches selon le vertical en pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- définissons 0 comme une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement de la couleur, de la taille, de la confus
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le bloc du calcul des valeurs de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- la boucle du calcul
   for(int i=1;i<rates_total;i++)
     {
      //--- si le prix courant Close est plus grand que le précédent, mettons la flèch
      if(close[i]>close[i-1])
         ArrowsBuffer[i]=close[i];
      //--- dans le cas contraire spécifions la valeur nulle
      else
         ArrowsBuffer[i]=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| modifie l'apparence des symboles dans l'indicateur               |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de l'indicateur
   string comm="";
//--- le bloc du changement de la couleur de la flèche
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de la taille des flèches
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // on spécifie la taille de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons la taille des flèches
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du code de la flèche (PLOT_ARROW)
   number=MathRand();
//---recevrons le reste de la division entière pour le calcul du nouveau code de la fl
   int code_add=number%20;
//---définissons un nouveau code du symbole comme la somme code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- inscrirons le code du symbole PLOT_ARROW
```

© 2000-2015, MetaQuotes Software Corp.

```
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- le bloc du changement du décalage des flèches selon la verticale en pixels
   number=MathRand();
//--- recevrons le décalage comme le reste de la division entière
   int shift=20-number%41;
//--- définissons un nouveau décalage de-20 jusqu'à 20
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- inscrirons le décalage PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_ZIGZAG

Le style DRAW_ZIGZAG dessine par la couleur spécifiée les segments selon les valeurs de deux tampons d'indicateur. Ce style est très semblable auDRAW_SECTION, mais à la différence du dernier permet de dessiner les segments verticaux dans la limite d'une barre, si on a spécifié les valeurs pour les deux tampons d'indicateur pour cette barre. Les segments se dessinent de la valeur dans un premier tampon jusqu'à la valeur dans un deuxième tampon d'indicateur. Aucun des tampons ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la ligne comme pour le style DRAW_SECTION - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

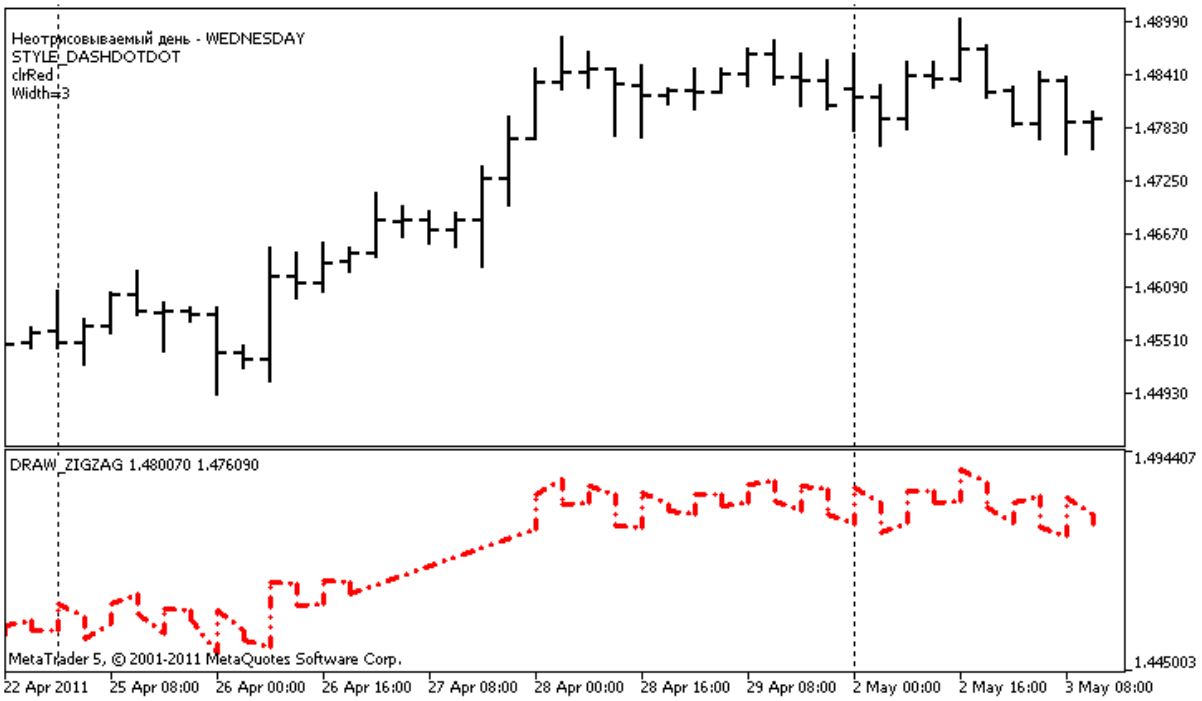
Les sections se dessinent d'une valeur non vide à l'autre valeur non vide du tampon d'indicateur. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriétéPLOT_EMPTY_VALUE:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(l'index_de la construction_DRAW_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

Le nombre de tampons nécessaires pour la construction DRAW_ZIGZAG — 2.

L'exemple de l'indicateur dessinant la scie de long selon les prix High et Low. La couleur, l'épaisseur et le style des lignes du zigzag se changent par hasard chaque **N** ticks.



Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_ZIGZAG on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() ces trois propriétés sont spécifiées par hasard. Le paramètre N est sorti dans les

© 2000-2015, MetaQuotes Software Corp.

paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                                  DRAW_ZIGZAG.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_ZIGZAG"
#property description "Dessine par les segments directs \"la scie \", en manquant les
#property description "Le jour des manques est choisi par hasard au lancement de l'inc
#property description "La couleur, l'épaisseur et le style des segments se changent"
#property description " par hasard par chaque N ticks"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ZigZag
#property indicator_label1  "ZigZag"
#property indicator_type1   DRAW_ZIGZAG
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres input
input int      N=5;              // le nombre de ticks pour le changement
//--- indicator buffers
double         ZigZagBuffer1[];
double         ZigZagBuffer2[];
//--- le jour de la semaine, pour lequel l'indicateur n'est pas dessiné
int invisible_day;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen};
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- l'enchaînement des tableaux et des tampons d'indicateur
   SetIndexBuffer(0,ZigZagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ZigZagBuffer2,INDICATOR_DATA);
//--- recevrons le nombre accidentel de 0 jusqu'à 6, pour ce jour l'indicateur n'est p
   invisible_day=MathRand()%6;
//---la valeur 0 (la valeur vide) ne participera pas au rendu
```

© 2000-2015, MetaQuotes Software Corp.

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetString(0,PLOT_LABEL,"ZigZag1;ZigZag2");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- la structure du temps sera nécessaire pour la réception du jour de la semaine de
   MqlDateTime dt;

//--- la position du commencement des calculs
   int start=0;
//---si l'indicateur a été calculé sur le tick précédent, nous commençons le calcul pa
   if(prev_calculated!=0) start=prev_calculated-1;
//--- la boucle des calculs
   for(int i=start;i<rates_total;i++)
     {
      //--- inscrirons le temps de l'ouverture de la barre dans la structure
      TimeToStruct(time[i],dt);
      //--- si le jour de la semaine cette barre est égal à invisible_day
      if(dt.day_of_week==invisible_day)
        {
         //--- inscrirons les valeurs vides aux tampons pour cette barre
         ZigZagBuffer1[i]=0;
```

© 2000-2015, MetaQuotes Software Corp.

```
         ZigZagBuffer2[i]=0;
        }
      //--- si le jour de la semaine est convenable, remplissons les tampons
      else
        {
         //--- si le numéro de la barre est pair
         if(i%2==0)
           {
            //---   écrivons au 1-er tampon High, et au 2-ème Low
            ZigZagBuffer1[i]=high[i];
            ZigZagBuffer2[i]=low[i];
           }
         //--- le numéro de la barre est impair
         else
           {
            //--- remplissons la barre dans l'ordre inverse
            ZigZagBuffer1[i]=low[i];
            ZigZagBuffer2[i]=high[i];
           }
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| modifie l'apparence des segments dans le zigzag                  |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de l'indicateur
   string comm="";
//--- le bloc du changement de la couleur du zigzag
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la ligne
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
```

© 2000-2015, MetaQuotes Software Corp.

```
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//---ajoutons l'information sur le jour, qui est manqué dans les calculs
   comm="\r\nLe jour non-dessiné - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+cor
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_FILLING

Le style DRAW_FILLING dessine le domaine coloré entre les valeurs de deux tampons d'indicateur. En fait ce style dessine deux lignes et peint l'espace entre eux par une de deux couleurs spécifiées. Il est destiné à la création des indicateurs dessinant les canaux. Aucun des tampons ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

On peut spécifier deux couleurs de remplissage

- la première couleur pour ces domaines, où les valeurs dans le premier tampon d'indicateur sont plus grandes que les valeurs dans le deuxième tampon d'indicateur;
- la deuxième couleur pour ces domaines, où les valeurs dans le deuxième tampon d'indicateur sont plus grandes que les valeurs dans le premier tampon d'indicateur.

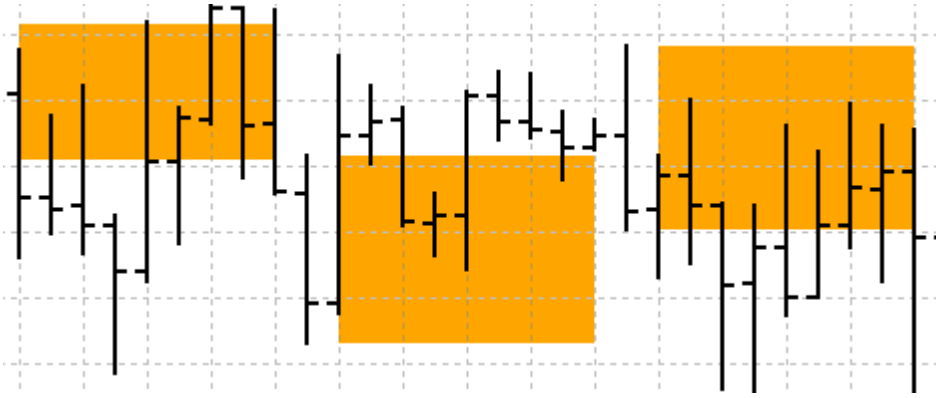
Vous pouvez spécifier la couleur de remplissagepar les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de "vivifier" les indicateurs pour qu'ils changent l'aspect en fonction de la situation actuelle.

L'indicateur est calculé pour toutes les barres, pour lesquelles les valeurs des deux tampons d'indicateur ne sont pas égales à 0 et ne sont pas égales à la valeur nulle. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriétéPLOT_EMPTY_VALUE:

```
   #define INDICATOR_EMPTY_VALUE -1.0
   ...
//--- la valeur INDICATOR_EMPTY_VALUE (une valeur vide) ne participera pas dans le cal
   PlotIndexSetDouble(L'indice_de la construction_DRAW_FILLING,PLOT_EMPTY_VALUE,INDICA
```

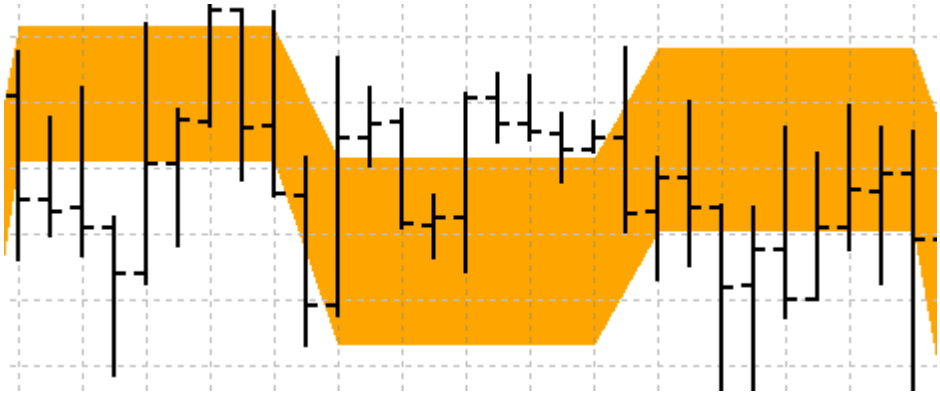
La rendu sur les barres, qui ne participent pas au calcul de l'indicateur, dépendra des valeurs dans les tampons d'indicateur:

- Les barres pour lesquelles les valeurs des deux tampons d'indicateurs sont égales à 0, ne participent pas à la rendu de l'indicateur. C'est-à-dire le domaine avec les valeurs nulles ne sera pas coloré.



- Les barres pour lesquelles les valeurs des deux tampons d'indicateurs sont égales à la valeur vide, ne participent pas à la rendu de l'indicateur. Le domaine avec les valeurs vides sera coloré de telle manière pour joindre les domaines avec les valeurs significatives.

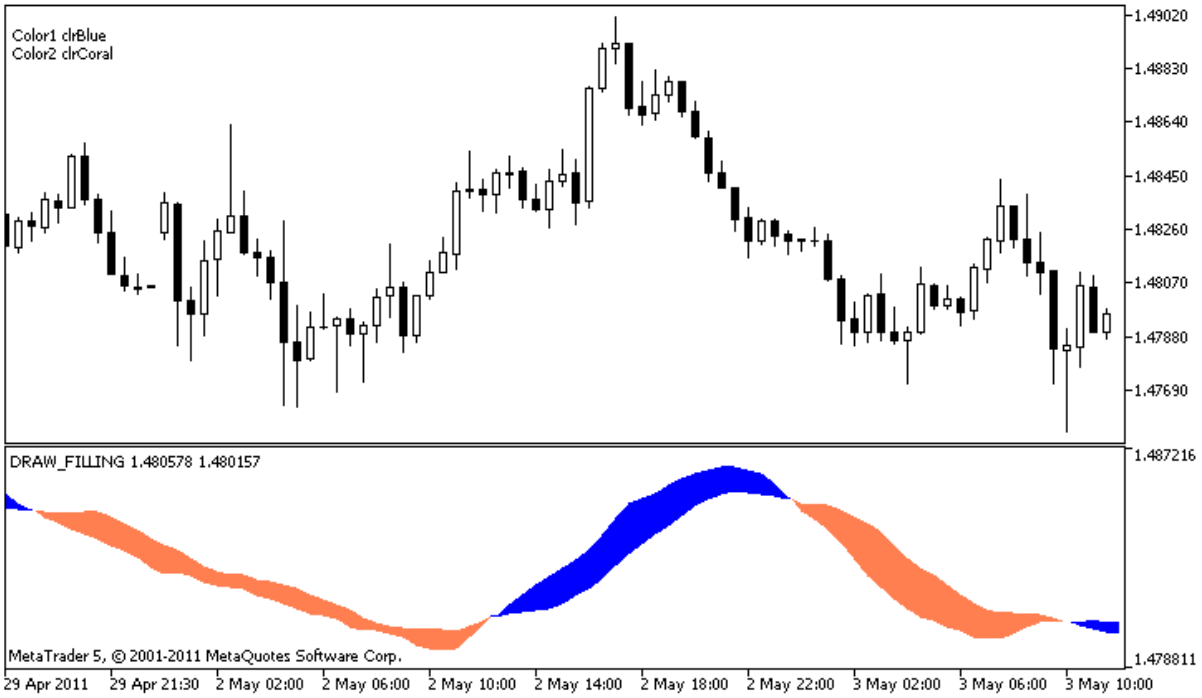
© 2000-2015, MetaQuotes Software Corp.



Il est important de marquer que si "une valeur vide" est égale au zéro, les barres qui ne participant pas dans le calcul de l'indicateur seront aussi colorées.

Le nombre de tampons nécessaires pour la construction DRAW_FILLING — 2.

L'exemple de l'indicateur dessinant dans la fenêtre séparée le canal entre deux glissants moyens avec différentes périodes de la prise de moyenne. Le changement de la couleur à l'intersection des moyens montre visuellement le remplacement des tendances montante et descendante. Les couleurs se changent chaques**N** ticks par hasard. Le paramètre N est porté auxparamètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans les propriétés de l'indicateur).



Prêtez attention que primordialement à la construction graphique**plot1** avec le style DRAW_FILLING on spécifie deux couleurs à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() les nouvelles couleurs sont spécifiées par hasard.

```
//+------------------------------------------------------------------+
//|                                                 DRAW_FILLING.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_FILLING"
#property description "Dessine dans la fenêtre séparée le canal entre deux moyens"
#property description "La couleur du remplissage du canal se change par hasard"
#property description "dans chaques N ticks"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot Intersection
#property indicator_label1  "Intersection"
#property indicator_type1   DRAW_FILLING
#property indicator_color1  clrRed,clrBlue
#property indicator_width1  1
//--- les paramètres input
input int      Fast=13;          // la période de la glissante moyenne rapide
input int      Slow=21;          // la période de la glissante moyenne lente
input int      shift=1;          // le décalage des moyennes au futur (positif)
input int      N=5;              // le nombre de ticks pour le changement
//--- les tampons d'indicateur
double         IntersectionBuffer1[];
double         IntersectionBuffer2[];
int fast_handle;
int slow_handle;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen,clrAquamarine,clrBlanchedAlmond,clrBrown,clrC
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,IntersectionBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,IntersectionBuffer2,INDICATOR_DATA);
//---
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//---
   fast_handle=iMA(_Symbol,_Period,Fast,0,MODE_SMA,PRICE_CLOSE);
   slow_handle=iMA(_Symbol,_Period,Slow,0,MODE_SMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseu
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//---faisons le premier calcul de l'indicateur ou les données sont changées et on dema
   if(prev_calculated==0)
     {
      //--- copions toutes les valeurs des indicateurs dans les tampons correspondants
      int copied1=CopyBuffer(fast_handle,0,0,rates_total,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,rates_total,IntersectionBuffer2);
     }
   else // remplissons économiquement seulement les données, qui sont mises à jour
     {
      //--- recevrons la différence dans les barres entre le lancement actuel et précé
      int to_copy=rates_total-prev_calculated;
      //--- s'il n'y a pas de différence, en tout cas copions une seule valeur - sur l
      if(to_copy==0) to_copy=1;
      //--- copions to_copy des valeurs à la fin de tampons d'indicateur
      int copied1=CopyBuffer(fast_handle,0,0,to_copy,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,to_copy,IntersectionBuffer2);
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| change les couleurs du remplissage du canal                      |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
```

© 2000-2015, MetaQuotes Software Corp.

```
   string comm="";
//--- le bloc du changement de la couleur de la ligne
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);

//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index1=number%size;
//--- définissons la première couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,colors[color_index1]);
//--- inscrirons la première couleur
   comm=comm+"\r\nColor1 "+(string)colors[color_index1];

//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   number=MathRand(); // recevrons le nombre accidentel
   int color_index2=number%size;
//--- définissons la deuxième couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,colors[color_index2]);
//--- inscrirons la deuxième couleur
   comm=comm+"\r\nColor2 "+(string)colors[color_index2];
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_BARS

Le style DRAW_BARS dessine les barres selon les valeurs de quatre tampons d'indicateur, qui contiennent les prix Open, High, Low et Close. Il est destiné à la création des indicateurs personnels en forme des barres, y compris dans une sous- fenêtre séparée du graphique et selon d'autres instruments financiers.

On peut spécifier la couleur des barres par les directives du compilateur ou dynamiquement à l'aide de la fonctionPlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de "vivifier" les indicateurs pour qu'ils changent l'aspect en fonction de la situation actuelle.

L'indicateur est dessiné seulement pour ces barres, pour lesquelles on a spécifié les valeurs non vides de **tous** les quatre tampons d'indicateur. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriété PLOT_EMPTY_VALUE:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(индекс_построения_DRAW_BARS,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

Le nombre de tampons nécessaires pour la construction DRAW_BARS — 4. Tous les tampons pour la construction doivent aller successivement un après l'autre en ordre donné: Open, High, Low et Close. Aucun des tampons ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

L'exemple de l'indicateur dessinant les barres selon l'instrument financier indiqué dans la fenêtre séparée. La couleur des barres se change chaques **N** ticks par hasard. Le paramètre N est porté auxparamètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans les propriétés de l'indicateur).

© 2000-2015, MetaQuotes Software Corp.

Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_BARS on spécifie la couleur à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() la couleur est spécifiée par hasard . par hasard de la liste d'avance préparée.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_BARS.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_BARS"
#property description "Dessine dans la fenêtre séparée les barres selon le symbole cho
#property description "La couleur et l'épaisseur et le symbole des barres se changent
#property description "par chaque N ticks"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plot Bars
#property indicator_label1  "Bars"
#property indicator_type1   DRAW_BARS
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres input
input int      N=5;              // le nombre de ticks pour le changement de l'aspect
input int      bars=500;         // combien de barres montrer
input bool     messages=false;   // l'affichage des messages dans le journal "Experts"
//--- les tampons d'indicateur
double         BarsBuffer1[];
double         BarsBuffer2[];
double         BarsBuffer3[];
double         BarsBuffer4[];
//--- le nom du symbole
string symbol;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- s'il y a trop peu de barres - terminons le travail avant terme
   if(bars<50)
     {
      Comment("Spécifiez un plus grand nombre de barres! Le travail de l'indicateur es
```

© 2000-2015, MetaQuotes Software Corp.

```
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,BarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,BarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,BarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,BarsBuffer4,INDICATOR_DATA);
//---le nom du symbole, selon lequel les barres se dessinent
   symbol=_Symbol;
//--- établissons l'affichage du symbole
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+symbol+")");
//--- une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseu
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- choisirons un nouveau symbole de la fenêtre "Aperçu du marché"
      symbol=GetRandomSymbolName();
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();

      int tries=0;
      //--- faisons 5 tentatives de remplir les tampons par les prix du symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total) && tries<5)
        {
         //--- le compteur des appels de la fonction CopyFromSymbolToBuffers()
         tries++;
        }
```

© 2000-2015, MetaQuotes Software Corp.

```
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur par les prix                |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total)
  {
//--- copions les prix Open, High, Low et Close au tableau rates[]
   MqlRates rates[];
//--- le compteur des tentatives
   int attempts=0;
//--- combien on a été copié
   int copied=0;
//--- faisons 25 tentatives de recevoir la série temporelle selon le symbole nécessai
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- si on n'a pas réussi à copier le nombre suffisant des barres
   if(copied!=bars)
     {
      //--- formons la chaîne du message
      string comm=StringFormat("On a réussi à recevoir seulement %d barres de %d barre
                               name,
                               copied,
                               bars
                               );
      //--- déduisons le message au commentaire sur une principale fenêtre du graphiqu
      Comment(comm);
      //--- déduisons les messages
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- établissons l'affichage du symbole
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+name+")");
     }
//--- initialisons les tampons par les valeurs vides
   ArrayInitialize(BarsBuffer1,0.0);
   ArrayInitialize(BarsBuffer2,0.0);
   ArrayInitialize(BarsBuffer3,0.0);
```

© 2000-2015, MetaQuotes Software Corp.

```
   ArrayInitialize(BarsBuffer4,0.0);
//--- copions les prix aux tampons
   for(int i=0;i<copied;i++)
     {
      //--- calculons l'index correspondant pour les tampons
      int buffer_index=total-copied+i;
      //--- inscrivons les prix aux tampons
      BarsBuffer1[buffer_index]=rates[i].open;
      BarsBuffer2[buffer_index]=rates[i].high;
      BarsBuffer3[buffer_index]=rates[i].low;
      BarsBuffer4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Rend accidentellement le symbole du Market Watch                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- le nombre de symboles montrés dans la fenêtre "Aperçu du marché"
   int symbols=SymbolsTotal(true);
//--- la position du symbole dans la liste - le nombre accidentel de 0 jusqu'aux symbo
   int number=MathRand()%symbols;
//--- rendons le nom du symbole selon la position indiquée
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence des barres                                   |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés des barres
   string comm="";
//--- le bloc du changement de la couleur des barres
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur de la ligne
   comm=comm+"\r\n"+(string)colors[color_index];

//--- le bloc du changement de l'épaisseur de la barre
   number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
```

© 2000-2015, MetaQuotes Software Corp.

```
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- inscrirons le nom du symbole
   comm="\r\n"+symbol+comm;

//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_CANDLES

Le style DRAW_CANDLES dessine les chandeliers japonais selon les valeurs de quatre tampons d'indicateur, qui contiennent les prix Open, High, Low et Close. Il est destiné à la création des indicateurs personnels en forme des bougies, y compris dans une sous- fenêtre séparée du graphique et selon d'autres financiers financiers.

On peut spécifier la couleur des barres par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de "vivifier" les indicateurs pour qu'ils changent l'aspect en fonction de la situation actuelle.

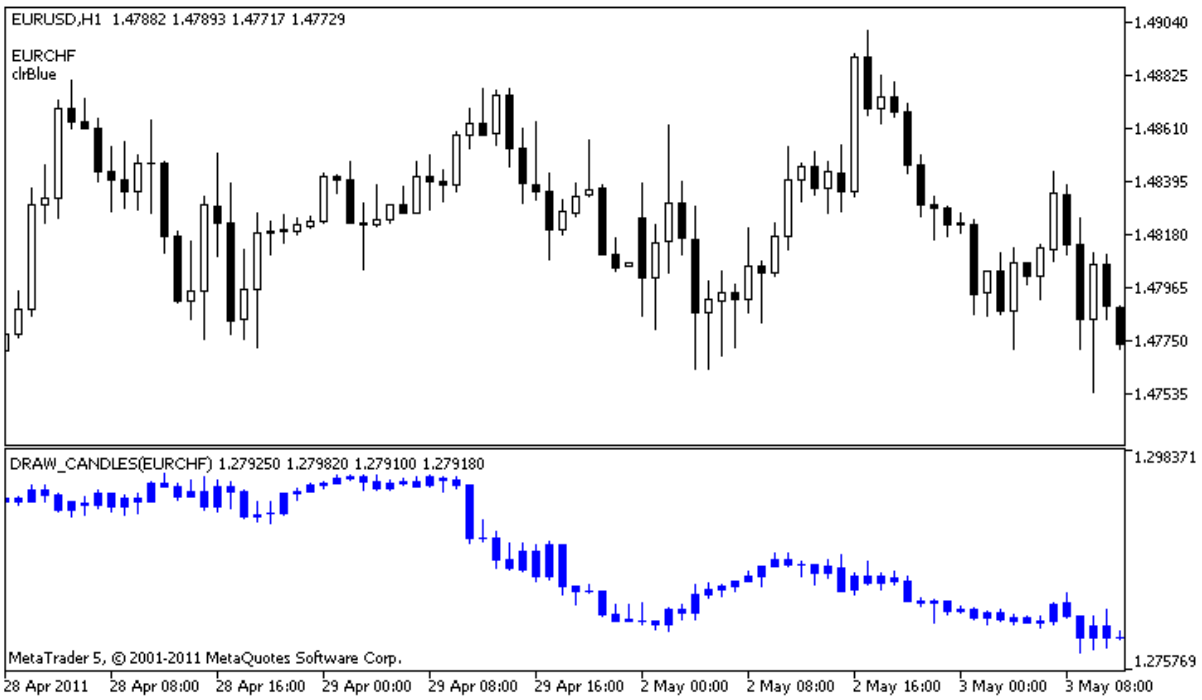
L'indicateur est dessiné seulement pour ces barres, pour lesquelles on a spécifié les valeurs non vides de **tous** les quatre tampons d'indicateur. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriété PLOT_EMPTY_VALUE:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(индекс_построения_DRAW_CANDLES,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

Le nombre de tampons nécessaires pour la construction DRAW_CANDLES — 4. Tous les tampons pour la construction doivent aller successivement un après l'autre en ordre donné: Open, High, Low et Close. Aucun des tampons ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

L'exemple de l'indicateur dessinant les chandeliers japonaises selon l'instrument financier indiqué dans la fenêtre séparée. La couleur des chandeliers se change chaques**N** ticks par hasard. Le paramètre N est porté auxparamètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans les propriétés de l'indicateur).



© 2000-2015, MetaQuotes Software Corp.

Prêtez attention que primordialement à la construction graphique **plot1**on spécifie la couleur à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() une nouvelle couleur est spécifiée par hasard de la liste d'avance préparée.

```
//+------------------------------------------------------------------+
//|                                                 DRAW_CANDLES.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_CANDLES."
#property description "Dessine dans la fenêtre séparée par les couleurs différentes le
#property description " "
#property description "La couleur et l'épaisseur et le symbole des bougies se changent
#property description "par hasard par chaque N ticks."

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plot Bars
#property indicator_label1  "DRAW_CANDLES1"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- les paramètres input
input int      N=5;              // le nombre de ticks pour le changement de l'aspect
input int      bars=500;         // combien de barres montrer
input bool     messages=false;   // l'affichage des messages dans le journal "Experts"
//--- les tampons d'indicateur
double         Candle1Buffer1[];
double         Candle1Buffer2[];
double         Candle1Buffer3[];
double         Candle1Buffer4[];
//--- le nom du symbole
string symbol;
//--- le tableau pour le stockage des couleurs
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- s'il y a trop peu de barres - terminons le travail avant terme
   if(bars<50)
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       Comment("Spécifiez un plus grand nombre de barres! Le travail de l'indicateur es
       return(INIT_PARAMETERS_INCORRECT);
      }
//--- indicator buffers mapping
   SetIndexBuffer(0,Candle1Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Candle1Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Candle1Buffer3,INDICATOR_DATA);
   SetIndexBuffer(3,Candle1Buffer4,INDICATOR_DATA);
//--- une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---le nom du symbole, selon lequel les barres se dessinent
   symbol=_Symbol;
//--- établissons l'affichage du symbole
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_CANDLES("+symbol+")");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseu
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- choisirons un nouveau symbole de la fenêtre "Aperçu du marché"
      symbol=GetRandomSymbolName();
      //--- remplaçons l'aspect
      ChangeLineAppearance();
      //--- choisirons un nouveau symbole de la fenêtre "Aperçu du marché"
      int tries=0;
      //--- faisons 5 tentatives de remplir les tampons plot1 par les prix du symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            Candle1Buffer1,Candle1Buffer2,Candle1Buffer3,Candle1Buffer4)
            && tries<5)
```

© 2000-2015, MetaQuotes Software Corp.

```
        {
         //--- le compteur des appels de la fonction CopyFromSymbolToBuffers()
         tries++;
        }
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplit une bougie indiquée                                      |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[]
                             )
  {
//--- copions les prix Open, High, Low et Close au tableau rates[]
   MqlRates rates[];
//--- le compteur des tentatives
   int attempts=0;
//--- combien on a été copié
   int copied=0;
//--- faisons 25 tentatives de recevoir la série temporelle selon le symbole nécessai
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- si on n'a pas réussi à copier le nombre suffisant des barres
   if(copied!=bars)
     {
      //--- formons la chaîne du message
      string comm=StringFormat("On a réussi à recevoir seulement %d barres de %d barre
                               name,
                               copied,
                               bars
                               );
      //--- déduisons le message au commentaire sur une principale fenêtre du graphiqu
      Comment(comm);
      //--- déduisons les messages
      if(messages) Print(comm);
      return(false);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
    else
      {
       //--- établissons l'affichage du symbole
       PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
      }
//--- initialisons les tampons par les valeurs vides
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- copions les prix aux tampons sur chaque tick
   for(int i=0;i<copied;i++)
     {
      //--- calculons l'index correspondant pour les tampons
      int buffer_index=total-copied+i;
      //--- inscrivons les prix aux tampons
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Rend accidentellement le symbole du Market Watch                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- le nombre de symboles montrés dans la fenêtre "Aperçu du marché"
   int symbols=SymbolsTotal(true);
//--- la position du symbole dans la liste - le nombre accidentel de 0 jusqu'aux symbo
   int number=MathRand()%symbols;
//--- rendons le nom du symbole selon la position indiquée
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence des barres                                   |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés des barres
   string comm="";
//--- le bloc du changement de la couleur des barres
   int number=MathRand(); // recevrons le nombre accidentel
//--- le diviseur du nombre est égal au montant du tableau colors[]
   int size=ArraySize(colors);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int color_index=number%size;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- inscrirons la couleur
   comm=comm+"\r\n"+(string)colors[color_index];
//--- inscrirons le nom du symbole
   comm="\r\n"+symbol+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_COLOR_LINE

Le style DRAW_COLOR_LINE c'est la variante colorée du style DRAW_LINE, il dessine aussi la ligne selon les valeur du tampon d'indicateur. Mais ce style, comme tous les styles de couleur qui ont **COLOR**dans le titre, a un tampon supplémentaire spécial d'indicateur pour stocker l'index (le numéro) de la couleur du tableau spécial spécifié des couleurs. Ainsi, on peut spécifier la couleur de chaque endroit, si on indique les index de la couleur, par lesquels la ligne doit être dessinée sur une barre spécifiée.

On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la ligne comme par les directives du compilateur, et dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

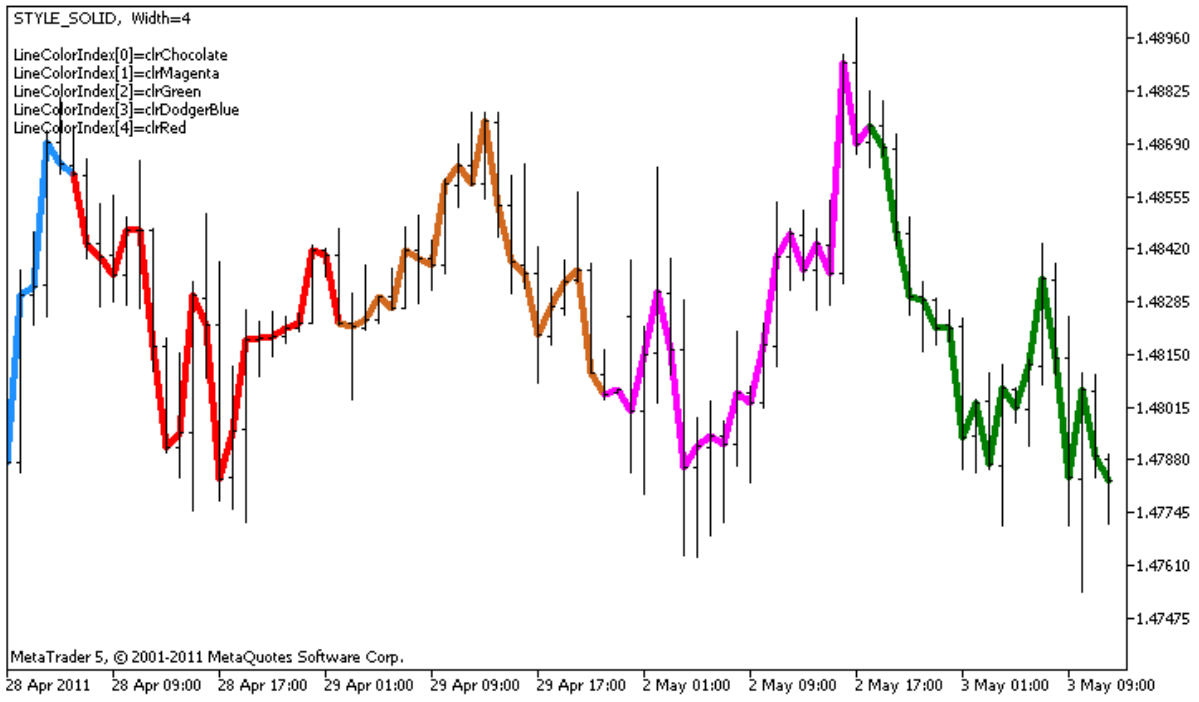
Le nombre de tampons nécessaires pour la construction DRAW_COLOR_LINE — 2:

- un tampon pour stocker la valeur de l'indicateur selon lequel se dessine la ligne;
- un tampon pour stocker l'index de la couleur, par lequel la ligne est dessinée sur chaque barre.

On peut spécifier les couleurs par la directive du compilateur #property indicator_color1 par la virgule. Le nombre de couleurs ne peut pas dépasser 64.

```
//--- spécifions 5 couleurs pour colorer chaque barre (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (on peut
```

L'exemple de l'indicateur dessinant la ligne aux prix de la clôture des barres Close. L'épaisseur et le style de la ligne se changent par hasard chaques N=5 ticks.



Les couleurs, par lesquelles les fragments de la ligne sont dessinés, se changent aussi par hasard dans la fonction d'utilisateur ChangeColors().

```
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Change la couleur des sections de la ligne                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
```

Dans l'exemple on montre la particularité des versions "de couleur" des indicateurs - pour changer la couleur du fragment de la ligne il ne faut pas changer la valeur dans le tampon ColorLineColors [] (qui contient les index des couleurs). Il suffit de spécifier les nouvelles couleurs dans un tableau spécial. Cela vous permet de changer rapidement la couleur pour toute la construction graphique, en changeant seulement un petit tableau des couleurs à l'aide de la fonctionPlotIndexSetInteger().

Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_COLOR_LINE on spécifie les propriétés à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() ces trois propriétés sont spécifiées par hasard.

Les paramètres N et Length (la longueur des fragments de couleur dans les barres) sont portés aux paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_LINE.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2015, MetaQuotes Software Corp.

```
#property description "L'indicateur pour la démonstration DRAW_COLOR_LINE"
#property description "Dessine la ligne par les morceaux colorés selon 20 barres aux p
#property description "La couleur, l'épaisseur et le style des fragments de la ligne s
#property description "par chaque N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
//--- spécifions 5 couleurs pour colorer chaque barre (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (on peut
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres input
input int      N=5;           // le nombre de ticks pour le changement
input int      Length=20;     // la longueur de chaque fragment de la couleur en barre
int            line_colors=5; // le nombre de couleurs spécifiées est égale à 5 - rega
//--- le tampon pour le dessin
double         ColorLineBuffer[];
//--- le tableau pour le stockage de la couleur du dessin de la ligne sur chaque barre
double         ColorLineColors[];

//--- le tableau pour le stockage des couleurs contient 7 éléments
color colors[]={clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGolde
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le binding du tableau et du tampon d'indicateur
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorLineColors,INDICATOR_COLOR_INDEX);
//--- l'initialisation du générateur des nombres pseudo-accidentels
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles les fragments de la ligne se dessin
      ChangeColors(colors,5);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le bloc du calcul des valeurs de l'indicateur
   for(int i=0;i<rates_total;i++)
     {
      //--- inscrirons la valeur de l'indicateur au tampon
      ColorLineBuffer[i]=close[i];
      //--- maintenant spécifions l'index de la couleur pour cette barre par hasard
      int color_index=i%(5*Length);
      color_index=color_index/Length;
      //--- pour cette barre la ligne sera dessinée par la couleur, qui se trouve sous
      ColorLineColors[i]=color_index;
     }

//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Change la couleur des sections de la ligne                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence de la ligne affichée dans l'indicateur       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+" Width="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   int size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm=EnumToString(styles[style_index])+", "+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_COLOR_SECTION

Le style DRAW_COLOR_SECTION c'est une variante colorée du style DRAW_SECTION, mais contrairement à ce dernier, permet de dessiner chaque section par sa propre couleur. Le style DRAW_COLOR_SECTION, comme tous les styles de couleur ayant**COLOR**dans le titre, contient le tampon supplémentaire spécial d'indicateur pour stocker l'index (le numéro) de la couleur du tableau spécial spécifié des couleurs. Ainsi, on peut spécifier la couleur de chaque section, si on indique les index de la couleur pour la barre, sur laquelle tombe la fin de la section.

On peut spécifier l'épaisseur, la couleur et le style des segments comme pour le style DRAW_SECTION - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

Les sections se dessinent d'une valeur non vide à l'autre valeur non vide du tampon d'indicateur, les valeurs vides sont manquées. Pour indiquer, quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriété PLOT_EMPTY_VALUE. Par exemple, si l'indicateur doit se dessiner par les segments selon les valeurs non nulles, il faut spécifier la valeur nulle comme vide:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(индекс_построения_DRAW_COLOR_SECTION,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs tous les éléments du tampon d'indicateur, spécifiez aux éléments non-rendus la valeur vide.

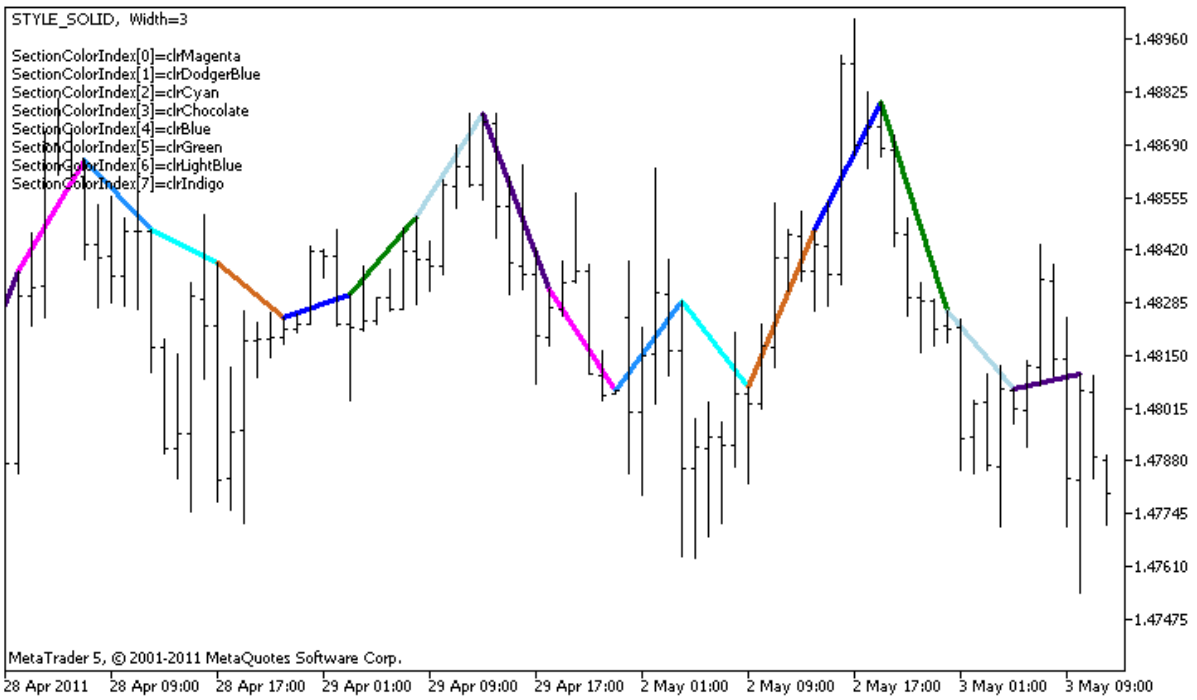
Le nombre de tampons nécessaires pour la construction DRAW_COLOR_SECTION — 2:

- un tampon pour stocker la valeur de l'indicateur selon lequel se dessine la ligne;
- un tampon pour stocker l'index de la couleur, par lequel la section est dessinée (il faut spécifier seulement pour les valeurs non vides).

On peut spécifier les couleurs par la directive du compilateur #property indicator_color1 par la virgule. Le nombre de couleurs ne peut pas dépasser 64.

L'exemple de l'indicateur dessinant les sections multicolores d'une longueur de 5 barres selon les prix High. La couleur, l'épaisseur et le style des sections se changent par hasard chaque**N** ticks.

© 2000-2015, MetaQuotes Software Corp.



Prêtez attention que primordialement à la construction graphique**plot1** avec le style DRAW_COLOR_SECTION on spécifie 8 couleurs à l'aide de la directive du compilateur #property. Puis dans la fonction OnCalculate() les couleurs sont spécifiées par hasard du tableau colors[].

Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_SECTION.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_SECTION"
#property description "Dessine par les segments colorés de la longueur au nombre spéci
#property description "La couleur, l'épaisseur et le style des sections se changent pa
#property description "dans chaques N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorSection
#property indicator_label1  "ColorSection"
#property indicator_type1   DRAW_COLOR_SECTION
//--- spécifions 8 couleurs pour colorer les sections (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrGold,clrMediumBlue,clrLime,clrMagenta,clrBrown,c
#property indicator_style1  STYLE_SOLID
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_width1  1
//--- les paramètres input
input int      N=5;                      // le nombre de ticks pour le changement
input int      bars_in_section=5;        // la longueur des sections en barres
//---la variable auxiliaire pour le calcul des fins des sections
int            divider;
int            color_sections;
//--- le tampon pour le dessin
double         ColorSectionBuffer[];
//--- le tableau pour le stockage de la couleur du dessin de la ligne sur chaque barre
double         ColorSectionColors[];
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorSectionBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorSectionColors,INDICATOR_COLOR_INDEX);
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- le nombre de couleurs pour colorier les sections
   color_sections=8;   //  regarder le commentaire pour la propriété #property indicat
//---vérifions le paramètre de l'indicateur
   if(bars_in_section<=0)
     {
      PrintFormat("La valeur inadmissible de la longueur la section=%d",bars_in_sectio
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=color_sections*bars_in_section;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles les sections sont dessinées
      ChangeColors(colors,color_sections);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le numéro de la barre, par laquelle nous commencerons le calcul des valeurs de l
   int start=0;
//--- si l'indicateur a été déjà calculé, établissons start sur la barre précédente
   if(prev_calculated>0) start=prev_calculated-1;
//--- Tous les calculs des valeurs de l'indicateur se trouvent ici
   for(int i=start;i<rates_total;i++)
     {
      //--- si le numéro de la barre se divise sans reste en longueur_de la section, c
      if(i%bars_in_section==0)
        {
         //---fixerons la fin du segment sur le prix High de cette barre
         ColorSectionBuffer[i]=high[i];
         //--- le reste de la division du numéro de la barre sur longueur_de la sectio
         int rest=i%divider;
         //recevrons le numéro de la couleur =  de 0 jusqu'au nombre_des couleurs-1
         int color_indext=rest/bars_in_section;
         ColorSectionColors[i]=color_indext;
        }
      //--- si le reste de la division est égal aux barres,
      else
        {
         //--- si rien n'est pas convenu, manquons cette barre - mettons la valeur 0
         ColorSectionBuffer[i]=0;
        }
     }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Change la couleur des sections de la ligne                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("SectionColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStr
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence de la ligne affichée dans l'indicateur       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+" Width="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   int size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm=EnumToString(styles[style_index])+", "+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_COLOR_HISTOGRAM

Le style DRAW_COLOR_HISTOGRAM dessine l'histogramme par les colonnes du zéro jusqu'à la valeur indiquée. Les valeurs sont tirées du tampon de l'indicateur. Chaque colonne peut avoir sa propre couleur de l'ensemble des couleurs d'avance prédéterminé.

n peut spécifier l'épaisseur, la couleur et le style du histogramme comme pour le style DRAW_HISTOGRAM – par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'histogramme en fonction de la situation actuelle.

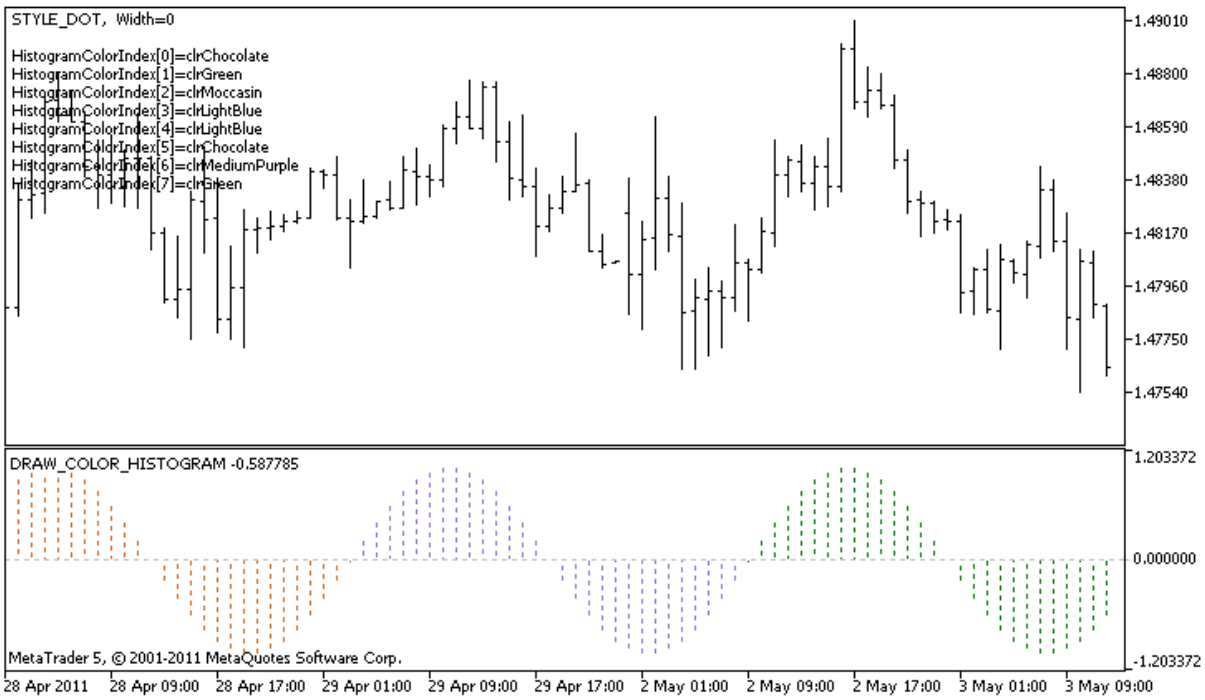
Puisque sur chaque barre se dessine la colonne du niveau nul, alors il est mieux d'utiliser DRAW_COLOR_HISTOGRAM pour l'affichage dans une sous- fenêtre séparée du graphique. Le plus souvent ce type de la construction graphique est utilisé pour la création des indicateurs du type d'oscillateur, par exemple, Awesome Oscillator ou Market Facilitation Index. Il suffit de spécifier la valeur nulle pour les valeurs vides non affichées.

Le nombre de tampons nécessaires pour la construction DRAW_COLOR_HISTOGRAM — 2:

- un tampon pour stocker de la valeur non nulle du segment vertical sur chaque barre, la deuxième fin du segment se trouve toujours sur la ligne zéro de l'indicateur;
- un tampon pour stocker l'index de la couleur, par lequel la section est dessinée (il faut spécifier seulement pour les valeurs non vides).

On peut spécifier les couleurs par la directive du compilateur #property indicator_color1 par la virgule. Le nombre de couleurs ne peut pas dépasser 64.

L'exemple de l'indicateur, dessinant la sinusoïde selon la fonctionMathSin()par la couleur spécifiée. La couleur, l'épaisseur et le style de**toutes** colonnes de l'histogramme se changent chaques **N** ticks par hasard. Le paramètre "bars" spécifie la période de la sinusoïde, c'est-à-dire dans le nombre spécifié de barres la sinusoïde se répétera selon la boucle.



© 2000-2015, MetaQuotes Software Corp.

Pretez attention que primordialement pour la construction graphique **plot1** avec le style DRAW_COLOR_HISTOGRAM on spécifie 5 couleurs à l'aide de la directive du compilateur#property indicator_color1, et puis dans la fonction OnCalculate() les couleurs sont choisies par hasard parmi 14 couleurs stockées dans le tableau colors[]. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_HISTOGRAM.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_HISTOGRAM"
#property description "Dessine la sinusoïde par l'histogramme dans la fenêtre séparée"
#property description "La couleur, l'épaisseur des colonnes se changent par hasard"
#property description "dans chaques N ticks"
#property description "Le paramètre "bars" spécifie le nombre de barres dans la boucle

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- les paramètres input
input int      bars=30;          // la période de la sinusoïde dans les barres
input int      N=5;              // le nombre de ticks pour le changement de l'histogr
//--- plot Color_Histogram
#property indicator_label1  "Color_Histogram"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
//--- spécifions 8 couleurs pour colorer les sections (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrGreen,clrBlue,clrYellow,clrMagenta,clrCyan,clrMe
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- le tampon des valeurs
double         Color_HistogramBuffer[];
//--- le tampon des index de la couleur
double         Color_HistogramColors[];
//--- le multiplicateur pour la réception de l'angle 2Pi dans les radians à la multipl
double         multiplier;
int            color_sections;
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- le tableau pour le stockage des styles du dessin de la ligne
```

© 2000-2015, MetaQuotes Software Corp.

```
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Color_HistogramBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,Color_HistogramColors,INDICATOR_COLOR_INDEX);
//---- le nombre de couleurs pour colorier la sinusoïde
   color_sections=8;   //  regarder le commentaire pour la propriété #property indicat
//--- calculerons le multiplicateur
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Spécifiez la valeur "bars"=%d plus que 1",bars);
      //--- une cessation anticipée du travail de l'indicateur
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseu
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles l'histogramme est dessiné
      ChangeColors(colors,color_sections);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- les calculs des valeurs de l'indicateur
   int start=0;
//--- si le calcul est déjà fait pendant le lancement précédent de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // établissons le début des calculs
//--- remplissons le tampon d'indicateur par les valeurs
   for(int i=start;i<rates_total;i++)
     {
      //--- la valeur
      Color_HistogramBuffer[i]=sin(i*multiplier);
      //--- la couleur
      int color_index=i%(bars*color_sections);
      color_index/=bars;
      Color_HistogramColors[i]=color_index;
     }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Change la couleur des sections de la ligne                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence de la ligne affichée dans l'indicateur       |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+" Width="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   int size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm=EnumToString(styles[style_index])+", "+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_COLOR_HISTOGRAM2

Le style DRAW_COLOR_HISTOGRAM2 dessine l'histogramme par la couleur spécifiée – les segments verticaux selon les valeurs de deux tampons d'indicateur. Mais à la différence de DRAW_HISTOGRAM2 monochrome on peut spécifier la couleur propre de l'ensemble prédéterminé dans ce style à chaque colonne de l'histogramme. Les valeurs des fins des segments sont tirées du tampon d'indicateur.

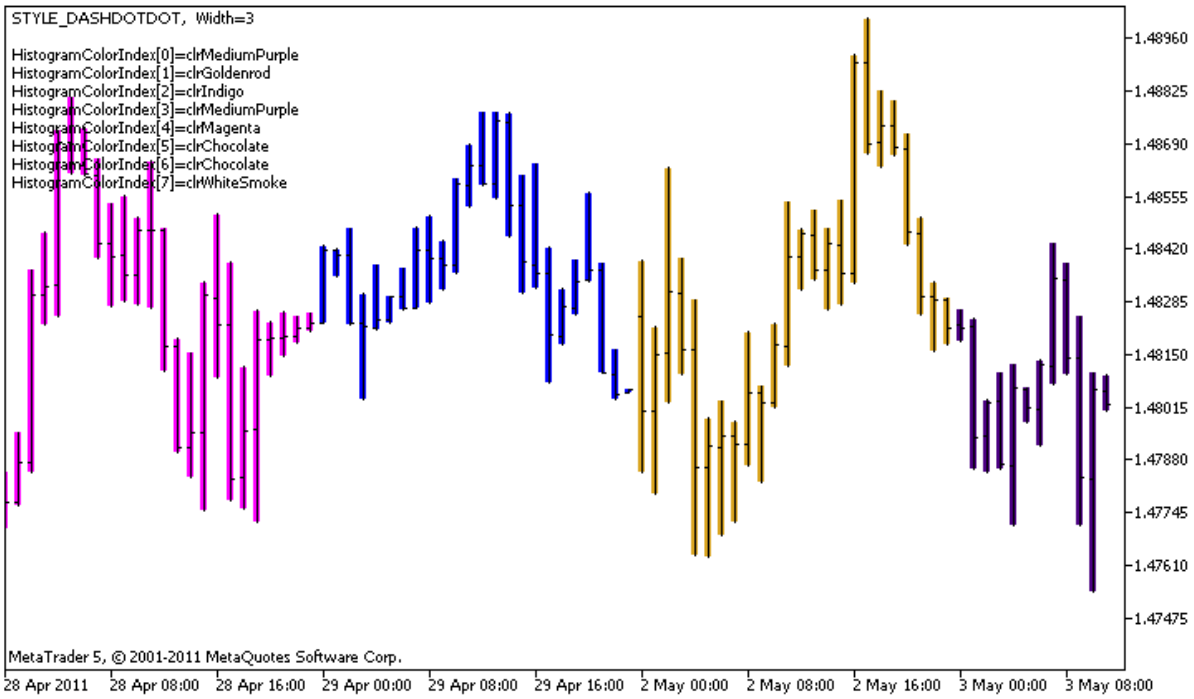
On peut spécifier l'épaisseur, le style et la couleur de l'histogramme comme pour le style DRAW_HISTOGRAM2 – par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'histogramme en fonction de la situation actuelle.

On peut utiliser le style DRAW_COLOR_HISTOGRAM2 dans une sous- fenêtre séparée du graphique, ainsi que dans une fenêtre principale. On ne fait pas le dessin pour les valeurs vides, toutes les valeurs dans les tampons d'indicateur doivent être installés de manière explicite. L'initialisation des tampons par la valeur vide n'est pas produite.

Le nombre de tampons nécessaires pour la construction DRAW_COLOR_HISTOGRAM2 — 3:

- deux tampons pour stocker l'extrémité supérieure et inférieure du segment vertical sur chaque barre;
- un tampon pour stocker l'index de la couleur, par lequel le segment est dessiné (il faut spécifier seulement pour les valeurs non vides).

L'exemple de l'indicateur, qui dessine l'histogramme entre les prix High et Low par la couleur spécifiée. Les lignes de l'histogramme se dessinent par sa couleur pour chaque jour de la semaine. La couleur de chaque jour, l'épaisseur et le style de l'histogramme se changent chaques**N** ticks par hasard.



Prêtez attention que primordialement à la construction graphique **plot1**avec le style DRAW_COLOR_HISTOGRAM2 on spécifie 5 couleurs à l'aide de la directive du compilateur #property indicator_color1, et puis dans la fonction OnCalculate()les couleurs sont choisies par hasard parmi 14 couleurs stockées dans le tableau colors[].

© 2000-2015, MetaQuotes Software Corp.

Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                        DRAW_COLOR_HISTOGRAM2.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_HISTOGRAM2"
#property description "Dessine le segment entre Open et Close sur chaque barre"
#property description "La couleur, l'épaisseur et le style se changent par hasard"
#property description "dans chaques N ticks"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plot ColorHistogram_2
#property indicator_label1  "ColorHistogram_2"
#property indicator_type1   DRAW_COLOR_HISTOGRAM2
//--- spécifions 5 couleurs pour colorer l'histogramme selon les jours de la semaine
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- le paramètre input
input int      N=5;              // le nombre de ticks pour le changement de l'histogr
int            color_sections;
//--- les tampons des valeurs
double         ColorHistogram_2Buffer1[];
double         ColorHistogram_2Buffer2[];
//--- le tampon des index de la couleur
double         ColorHistogram_2Colors[];
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorHistogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorHistogram_2Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorHistogram_2Colors,INDICATOR_COLOR_INDEX);
//---établissons la valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- le nombre de couleurs pour colorier la sinusoïde
   color_sections=8;   //  regarder le commentaire à la propriété #property indicator_
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles l'histogramme est dessiné
      ChangeColors(colors,color_sections);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- les calculs des valeurs de l'indicateur
   int start=0;
//--- pour la réception du jour de la semaine selon le temps de l'ouverture de chaque
   MqlDateTime dt;
//--- si le calcul est déjà fait pendant le lancement précédent de OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // établissons le début des calculs
//--- remplissons le tampon d'indicateur par les valeurs
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- la valeur
      ColorHistogram_2Buffer1[i]=high[i];
      ColorHistogram_2Buffer2[i]=low[i];
      //---spécifions l'index de la couleur par jour de la semaine
      int day=dt.day_of_week;
      ColorHistogram_2Colors[i]=day;
     }
//--- rendons la valeur prev_calculated pour un appel suivant de la fonction
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Change la couleur des sections de la ligne                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence de la ligne affichée dans l'indicateur       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+" Width="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   int size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm=EnumToString(styles[style_index])+", "+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_COLOR_ARROW

Le style DRAW_COLOR_ARROW dessine les flèches sur le graphique par la couleur (les symboles de l'ensemble Wingdings) selon la valeur du tampon d'indicateur. A la différence du style DRAW_ARROW, on y peut spécifier la couleur pour chaque symbole de l'ensemble des couleurs prédéterminé, spécifiées par la propriété indicator_color1.

On peut spécifier l'épaisseur et la couleur des symboles de la meme façon que pour le style DRAW_ARROW - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de changer l'aspect de l'histogramme en fonction de la situation actuelle.

Le code du symbole pour la sortie sur le graphique est spécifié à l'aide de la propriété PLOT_ARROW.

```
//--- - spécifions le code du symbole de l'ensemble Wingdings pour le dessin dans PLOT
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

Par défaut la valeur PLOT_ARROW=159 (le cercle).

Chaque flèche représente en réalité le symbole, qui a la hauteur et le point du rattachement, et peut fermer par lui-même une certaine importante information sur le graphique (par exemple, le prix de la clôture de la barre). C'est pourquoi on peut spécifier en supplément un décalage vertical en pixels, qui ne dépend pas de l'échelle du graphique. Les flèches seront visuellement déplacées selon la verticale sur le nombre indiqué de pixels, bien que les valeurs de l'indicateur restent les mêmes:

```
//--- spécifions le décalage des flèches selon le vertical en pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

La valeur négative PLOT_ARROW_SHIFT signifie le décalage des flèches en haut, la valeur positive déplace les flèches en bas.

On peut utiliser le style DRAW_COLOR_ARROW dans une sous- fenêtre séparée du graphique, ainsi que dans une fenêtre principale.. Les valeurs vides ne se dessinent pas et ne s'affichent pas dans "la Fenêtre des données", toutes les valeurs dans les tampons d'indicateur doivent être installés de manière explicite. L'initialisation des tampons par la valeur vide n'est pas produite.

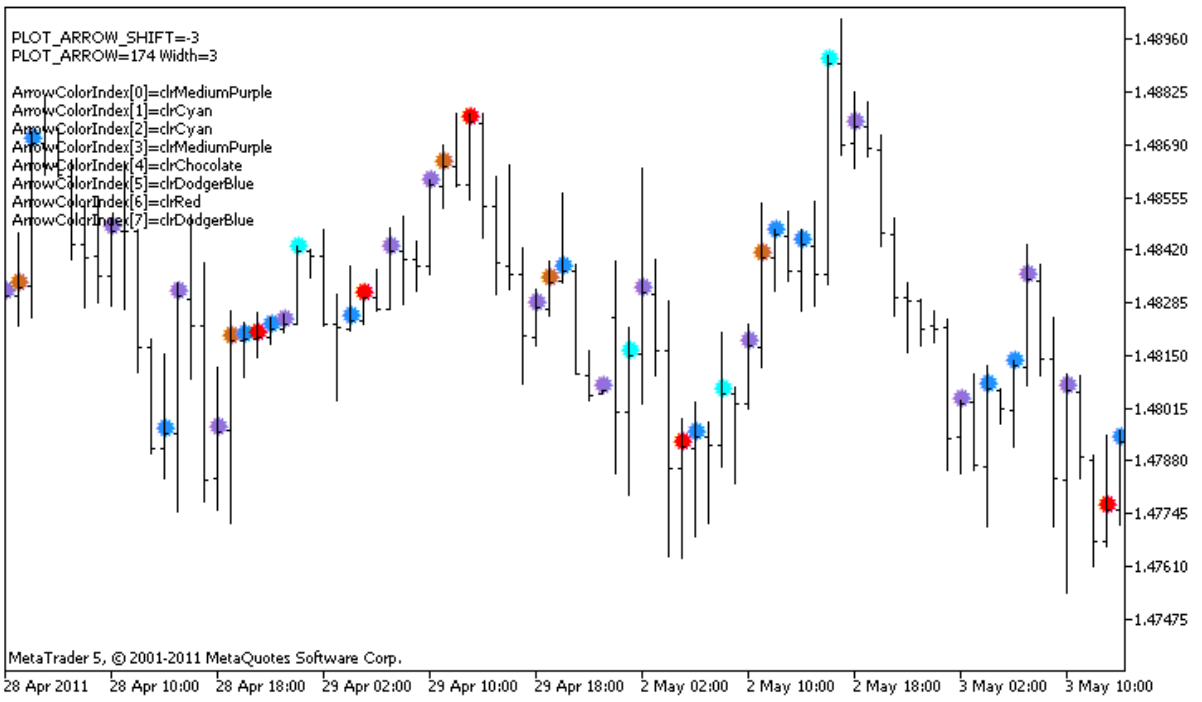
```
//---établissons la valeur vide
   PlotIndexSetDouble(l'index_de la construction_DRAW_COLOR_ARROW,PLOT_EMPTY_VALUE,0);
```

Le nombre de tampons nécessaires pour la construction DRAW_COLOR_ARROW — 2:

- un tampon pour stocker la valeur du prix selon lequel se dessine le symbole (en plus le décalage en pixels, spécifié par la propriété PLOT_ARROW_SHIFT);
- un tampon pour stocker l'index de la couleur, par lequel se dessine la flèche (il faut spécifier seulement pour les valeurs non vides).

L'exemple de l'indicateur dessinant les flèches sur chaque barre, qui a le prix de la clôture Close plus grand que le prix de la clôture de la barre précédente. L'épaisseur, le décalage et le code du symbole de**toutes** les flèches se changent chaques **N** ticks par hasard. La couleur du symbole dépend du numéro de la barre, sur lequelle il est dessiné.

© 2000-2015, MetaQuotes Software Corp.



Dans l'exemple les propriétés la couleur et la taille pour la construction graphique **plot1** avec le style DRAW_COLOR_ARROW sont spécifiés primordialement à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate()les propriétés sont spécifiées par hasard.. Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

Pretez attention que primordialement on spécifie 8 couleurs à l'aide de la directive du compilateur#property, et puis dans la fonctionOnCalculate() la couleur est choisie par hasard parmi 14 couleurs stockées dans le tableau colors[].

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_ARROW.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_ARROW"
#property description "Dessine sur le graphique par les différentes couleurs les flèch
#property description "La couleur, la taille, le décalage et le code du symbole de la
#property description " par hasard par chaque N ticks"
#property description "Le paramètre code spécifie la valeur de base: le code = 159 (ce

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorArrow
#property indicator_label1  "ColorArrow"
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_type1   DRAW_COLOR_ARROW
//--- spécifions 8 couleurs pour colorer l'histogramme selon les jours de la semaine
#property indicator_color1  clrRed,clrBlue,clrSeaGreen,clrGold,clrDarkOrange,clrMagent
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- les paramètres input
input int       N=5;          // le nombre de ticks pour le changement
input ushort    code=159;     // le code du symbole pour le dessin dans DRAW_ARROW
int             color_sections;
//--- le tampon d'indicateur pour la construction
double          ColorArrowBuffer[];
//--- le tableau pour le stockage des index de la couleur
double          ColorArrowColors[];
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorArrowBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorArrowColors,INDICATOR_COLOR_INDEX);
//--- spécifions le code du symbole pour le dessin dans PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- spécifions le décalage des flèches selon le vertical en pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- définissons 0 comme une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- le nombre de couleurs pour colorier la sinusoïde
   color_sections=8;   //  regarder le commentaire pour la propriété #property indicat
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement de la couleur, de la taille, de la confus
   ticks++;
//--- si le nombre critique des ticks est accumulé
   if(ticks>=N)
     {
      //--- changeons les propriétés des flèches
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles l'histogramme est dessiné
      ChangeColors(colors,color_sections);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- le bloc du calcul des valeurs de l'indicateur
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- la boucle du calcul
   for(int i=1;i<rates_total;i++)
     {
      //--- si le prix courant Close est plus grand que le précédent, mettons la flèch
      if(close[i]>close[i-1])
         ColorArrowBuffer[i]=close[i];
      //--- dans le cas contraire spécifions la valeur nulle
      else
         ColorArrowBuffer[i]=0;
      //--- la couleur de la flèche
      int index=i%color_sections;
      ColorArrowColors[i]=index;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Change la couleur des sections de la ligne                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       //--- recevrons le nombre accidentel
       int number=MathRand();
       //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
       int i=number%size;
       //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLO
       PlotIndexSetInteger(0,                    //  le numéro du style graphique
                           PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                           plot_color_ind,       //  l'index de la couleur, où nous ins
                           cols[i]);             //   une nouvelle couleur
       //--- inscrirons les couleurs
       comm=comm+StringFormat("ArrowColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStrin
       ChartSetString(0,CHART_COMMENT,comm);
      }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence de la ligne affichée dans l'indicateur       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés de la ligne
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5; // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+" Width="+IntegerToString(width);

//--- le bloc du changement du code de la flèche (PLOT_ARROW)
   number=MathRand();
//---recevrons le reste de la division entière pour le calcul du nouveau code de la fl
   int code_add=number%20;
//---définissons un nouveau code du symbole comme la somme code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- inscrirons le code du symbole PLOT_ARROW
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- le bloc du changement du décalage des flèches selon la verticale en pixels
   number=MathRand();
//--- recevrons le décalage comme le reste de la division entière
   int shift=20-number%41;
//--- définissons un nouveau décalage
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- inscrirons le décalage PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_COLOR_ZIGZAG

Le style DRAW_COLOR_ZIGZAG dessine les segments des couleurs différentes selon les valeurs de deux tampons d'indicateur. Ce style est la version colorée du style DRAW_ZIGZAG, c'est-à-dire permet de spécifier à chaque segment une propre couleur de l'ensemble des couleurs d'avance prédéterminé. Les segments se dessinent de la valeur dans un premier tampon jusqu'à la valeur dans un deuxième tampon d'indicateur. Aucun des tampons ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

On peut spécifier l'épaisseur, la couleur et le style de l'affichage de la ligne comme pour le style DRAW_ZIGZAG - par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de créer les indicateurs "vivants", qui changent leur aspect en fonction de la situation actuelle.

Les sections se dessinent d'une valeur non vide à l'autre valeur non vide du tampon d'indicateur. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriétéPLOT_EMPTY_VALUE:

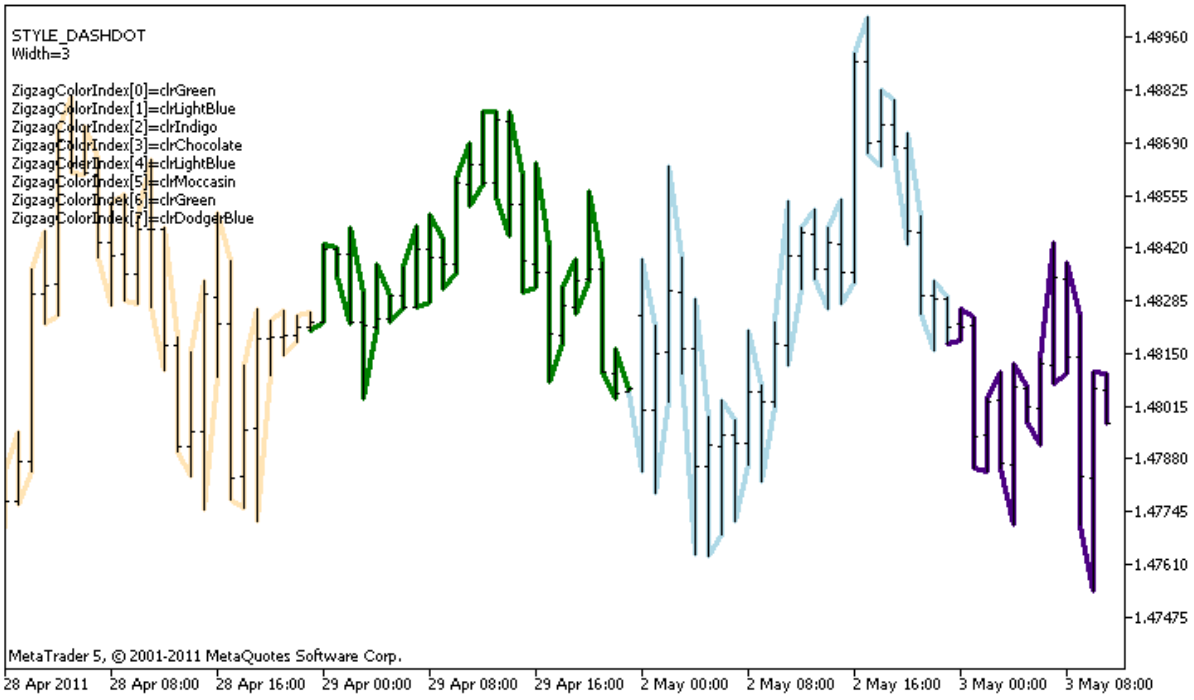
```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(l'index_de la construction_DRAW_COLOR_ZIGZAG,PLOT_EMPTY_VALUE,0)
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

Le nombre de tampons nécessaires pour la construction DRAW_COLOR_ZIGZAG — 3:

- deux tampons pour la conservation des valeurs des fins des sections du zigzag;
- un tampon pour stocker l'index de la couleur, par lequel la section est dessinée (il faut spécifier seulement pour les valeurs non vides).

L'exemple de l'indicateur dessinant la scie de long selon les prix High et Low. La couleur, l'épaisseur et le style de la ligne du zigzag se changent par hasard chaque**N** ticks.



© 2000-2015, MetaQuotes Software Corp.

Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_COLOR_ZIGZAG on spécifie 8 couleurs à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() la couleur est choisie par hasard parmi 14 couleurs stockées dans le tableau colors[].

Le paramètre N est sorti dans les paramètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans la fenêtre des propriétés de l'indicateur).

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_ZIGZAG.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_ZIGZAG"
#property description "Dessine la ligne brisée par les segments colorés, la couleur dé
#property description "La couleur, l'épaisseur et le style des segments se changent"
#property description " par hasard par chaque N ticks"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plot Color_Zigzag
#property indicator_label1  "Color_Zigzag"
#property indicator_type1   DRAW_COLOR_ZIGZAG
//--- spécifions 8 couleurs pour colorer les sections (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- le paramètre input
input int       N=5;              // le nombre de ticks pour le changement
int             color_sections;
//--- les tampons des valeurs des fins des segments
double          Color_ZigzagBuffer1[];
double          Color_ZigzagBuffer2[];
//--- le tampon des index de la couleur des fins des segments
double          Color_ZigzagColors[];
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- le tableau pour le stockage des styles du dessin de la ligne
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Color_ZigzagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Color_ZigzagBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Color_ZigzagColors,INDICATOR_COLOR_INDEX);
//---- le nombre de couleurs pour colorier le zigzag
   color_sections=8;   //  regarder le commentaire pour la propriété #property indicat
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- calculons les ticks pour le changement du style, de la couleur et de l'épaisseu
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles les sections sont dessinées
      ChangeColors(colors,color_sections);
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }

//--- la structure du temps sera nécessaire pour la réception du jour de la semaine de
   MqlDateTime dt;

//--- la position du commencement des calculs
   int start=0;
//---si l'indicateur a été calculé sur le tick précédent, nous commençons le calcul pa
   if(prev_calculated!=0) start=prev_calculated-1;
//--- la boucle des calculs
```

© 2000-2015, MetaQuotes Software Corp.

```
   for(int i=start;i<rates_total;i++)
     {
      //--- inscrirons le temps de l'ouverture de la barre dans la structure
      TimeToStruct(time[i],dt);

      //--- si le numéro de la barre est pair
      if(i%2==0)
        {
         //---  écrivons au 1-er tampon High, et au 2-ème - Low
         Color_ZigzagBuffer1[i]=high[i];
         Color_ZigzagBuffer2[i]=low[i];
         //--- la couleur du segment
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
      //--- le numéro de la barre est impair
      else
        {
         //--- remplissons la barre dans l'ordre inverse
         Color_ZigzagBuffer1[i]=low[i];
         Color_ZigzagBuffer2[i]=high[i];
         //--- la couleur du segment
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Change la couleur des segments du zigzag                         |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- inscrirons les couleurs
      comm=comm+StringFormat("ZigzagColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence des segments dans le zigzag                  |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés des Color_ZigZag
   string comm="";
//--- le bloc du changement de l'épaisseur de la ligne
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- le bloc du changement du style de la ligne
   number=MathRand();
//--- le diviseur du nombre est égal à la taille du tableau styles
   int size=ArraySize(styles);
//--- recevrons l'index pour le choix de la nouvelle couleur comme le reste de la divi
   int style_index=number%size;
//--- définissons la couleur comme la propriété PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- inscrirons le style de la ligne
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DRAW_COLOR_BARS

Le style DRAW_COLOR_BARS dessine les barres selon les valeurs de quatre tampons d'indicateur, qui contiennent les prix Open, High, Low et Close. Ce style est une version avancée du style DRAW_BARS et permet de spécifier la couleur de l'ensemble des couleurs d'avance prédéterminé pour chaque barre. Il est destiné à la création des indicateurs personnels en forme des barres, y compris dans une sous-fenêtre séparée du graphique et selon d'autres instruments financiers.

On peut spécifier la couleur des barres par les directives du compilateur ou dynamiquement à l'aide de la fonctionPlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de "vivifier" les indicateurs pour qu'ils changent l'aspect en fonction de la situation actuelle.

L'indicateur est dessiné seulement pour ces barres, pour lesquelles on a spécifié les valeurs non vides de **tous** les quatre tampons d'indicateur. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriété PLOT_EMPTY_VALUE:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(индекс_построения_DRAW_COLOR_BARS,PLOT_EMPTY_VALUE,0);
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

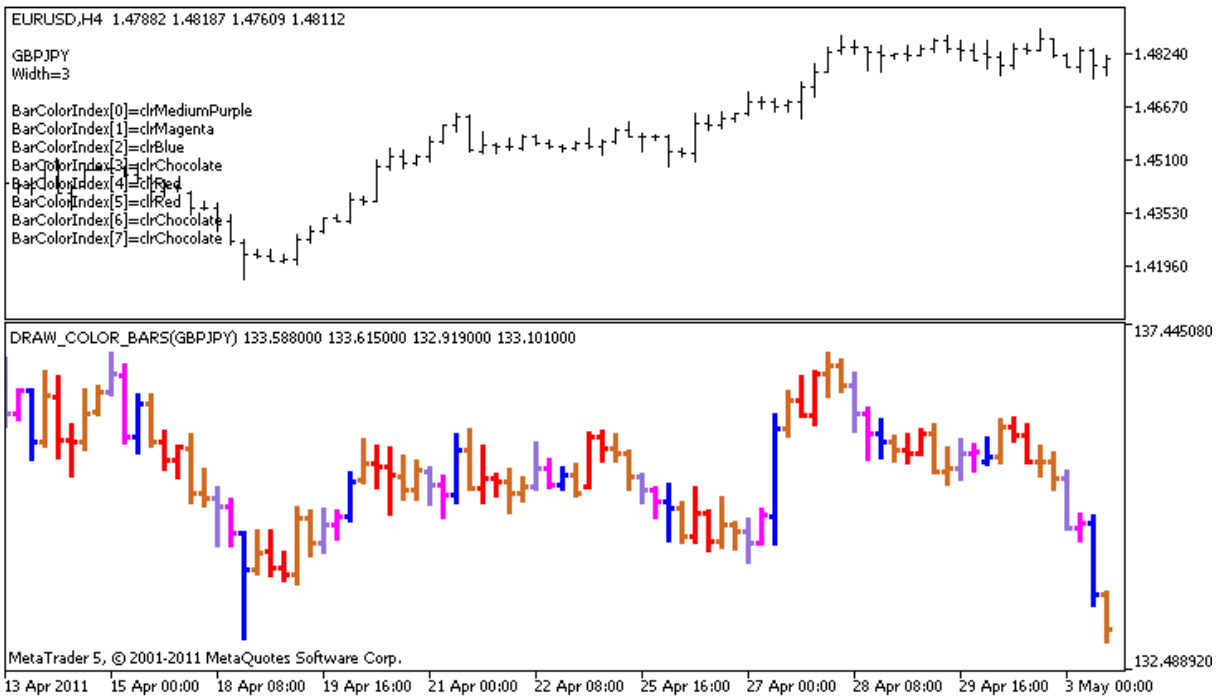
Le nombre de tampons nécessaires pour la construction DRAW_COLOR_BARS — 5:

- quatre tampons pour stocker les valeurs Open, High, Low et Close;
- un tampon pour stocker l'index de la couleur, par lequel la barre est dessinée (il faut spécifier seulement pour les barres dessinées).

Tous les tampons pour la construction doivent aller successivement un après l'autre en ordre donné: Open, High, Low, Close et le tampon de la couleur. Aucun des tampons de prix ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

L'exemple de l'indicateur dessinant les barres selon l'instrument financier indiqué dans la fenêtre séparée. La couleur des barres se change chaques **N** ticks par hasard. Le paramètre N est porté auxparamètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans les propriétés de l'indicateur).

© 2000-2015, MetaQuotes Software Corp.



Prêtez attention que primordialement à la construction graphique **plot1** avec le style DRAW_COLOR_BARS on spécifie 8 couleurs à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() les couleurs sont choisies par hasard parmi 14 couleurs stockées dans le tableau colors[].

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_BARS.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_BARS"
#property description "Dessine dans la fenêtre séparée par les couleurs différentes le
#property description "La couleur et l'épaisseur et le symbole des barres se changent
#property description "par chaque N ticks"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorBars
#property indicator_label1  "ColorBars"
#property indicator_type1   DRAW_COLOR_BARS
//--- spécifions 8 couleurs pour colorer les barres (ils se trouvent dans un tableau s
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les paramètres input
```

© 2000-2015, MetaQuotes Software Corp.

```
input int      N=5;              // le nombre de ticks pour le changement de l'aspect
input int      bars=500;         // combien de barres montrer
input bool     messages=false;   // l'affichage des messages dans le journal "Experts"
//--- les tampons d'indicateur
double         ColorBarsBuffer1[];
double         ColorBarsBuffer2[];
double         ColorBarsBuffer3[];
double         ColorBarsBuffer4[];
double         ColorBarsColors[];
//--- le nom du symbole
string symbol;
int    bars_colors;
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorBarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorBarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorBarsBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorBarsColors,INDICATOR_COLOR_INDEX);
//---- le nombre de couleurs pour colorier les barres
   bars_colors=8;   //  à voir le commentaire pour la propriété #property indicator_co
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- calculons les tics pour le changement du style, de la couleur et de l'épaisseur
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- choisirons un nouveau symbole de la fenêtre "Aperçu du marché"
      symbol=GetRandomSymbolName();
      //--- changeons les propriétés de la ligne
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles les barres sont dessinées
      ChangeColors(colors,bars_colors);
      int tries=0;
      //--- faisons 5 tentatives de remplir les tampons par les prix du symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,bars_colors) && tries<5)
        {
         //--- le compteur des appels de la fonction CopyFromSymbolToBuffers()
         tries++;
        }
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur par les prix                |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total,int bar_colors)
  {
//--- copions les prix Open, High, Low et Close au tableau rates[]
   MqlRates rates[];
//--- le compteur des tentatives
   int attempts=0;
//--- combien on a été copié
   int copied=0;
//--- faisons 25 tentatives de recevoir la série temporelle selon le symbole nécessai
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- si on n'a pas réussi à copier le nombre suffisant des barres
   if(copied!=bars)
     {
      //--- formons la chaîne du message
      string comm=StringFormat("On a réussi à recevoir seulement %d barres de %d barre
                               name,
                               copied,
```

© 2000-2015, MetaQuotes Software Corp.

```
                                bars
                                );
      //--- déduisons le message au commentaire sur une principale fenêtre du graphiqu
      Comment(comm);
      //--- déduisons les messages
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- établissons l'affichage du symbole
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_BARS("+name+")");
     }
//--- initialisons les tampons par les valeurs vides
   ArrayInitialize(ColorBarsBuffer1,0.0);
   ArrayInitialize(ColorBarsBuffer2,0.0);
   ArrayInitialize(ColorBarsBuffer3,0.0);
   ArrayInitialize(ColorBarsBuffer4,0.0);

//--- copions les prix aux tampons
   for(int i=0;i<copied;i++)
     {
      //--- calculons l'index correspondant pour les tampons
      int buffer_index=total-copied+i;
      //--- inscrivons les prix aux tampons
      ColorBarsBuffer1[buffer_index]=rates[i].open;
      ColorBarsBuffer2[buffer_index]=rates[i].high;
      ColorBarsBuffer3[buffer_index]=rates[i].low;
      ColorBarsBuffer4[buffer_index]=rates[i].close;
      //---
      ColorBarsColors[buffer_index]=i%bar_colors;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Rend accidentellement le symbole du Market Watch                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- le nombre de symboles montrés dans la fenêtre "Aperçu du marché"
   int symbols=SymbolsTotal(true);
//--- la position du symbole dans la liste - le nombre accidentel de 0 jusqu'aux symbo
   int number=MathRand()%symbols;
//--- rendons le nom du symbole selon la position indiquée
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Change la couleur des segments du zigzag                         |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("BarColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence des barres                                   |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés des barres
   string comm="";

//--- le bloc du changement de l'épaisseur de la barre
   int number=MathRand();
//--- recevrons l'épaisseur comme le reste de la division entière
   int width=number%5;   // l'épaisseur est spécifiée de 0 jusqu'à 4
//--- définissons la couleur comme la propriété PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- inscrirons l'épaisseur de la ligne
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- inscrirons le nom du symbole
   comm="\r\n"+symbol+comm;

//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

DRAW_COLOR_CANDLES

Le style DRAW_COLOR_CANDLES comme DRAW_CANDLES dessine les chandeliers japonais selon les valeurs de quatre tampons d'indicateur, qui contiennent les prix Open, High, Low et Close. Mais outre cela il permet de spécifier la couleur pour chaque bougie de l'ensemble donné. Pour cela on a ajouté le tampon de couleur spécial au style, qui garde les index des couleurs pour chaque barre. Il est destiné à la création des indicateurs personnels en forme des bougies, y compris dans une sous- fenêtre séparée du graphique et selon d'autres financiers financiers.

On peut spécifier le nombre de couleurs pour le coloriage des bougies par les directives du compilateur ou dynamiquement à l'aide de la fonction PlotIndexSetInteger(). Le changement dynamique des propriétés de la construction graphique permet de "vivifier" les indicateurs pour qu'ils changent l'aspect en fonction de la situation actuelle.

L'indicateur est dessiné seulement pour ces barres, pour lesquelles on a spécifié les valeurs non vides de quatre tampons d'indicateur pour garder les prix. Pour spécifier quelle valeur doit être considérée comme "vide", établissez cette valeur dans la propriétéPLOT_EMPTY_VALUE:

```
//---la valeur 0 (la valeur vide) ne participera pas au rendu
   PlotIndexSetDouble(l'index_de la construction_DRAW_COLOR_CANDLES,PLOT_EMPTY_VALUE,(
```

Remplissez toujours évidemment par les valeurs les tampons d'indicateur, indiquez une valeur vide dans le tampon sur les barres manquées.

Le nombre de tampons nécessaires pour la construction DRAW_COLOR_CANDLES — 5:

- quatre tampons pour stocker les valeurs Open, High, Low et Close;
- le tampon pour stocker l'index de la couleur, par lequel la bougie est dessinée (il est logique de spécifier seulement les bougies dessinées).

Tous les tampons pour la construction doivent aller successivement un après l'autre en ordre donné: Open, High, Low, Close et le tampon de la couleur. Aucun des tampons de prix ne peut pas contenir seulement les valeurs vides, puisque dans ce cas le dessin ne passe pas.

L'exemple de l'indicateur dessinant les chandeliers japonaises selon l'instrument financier indiqué dans la fenêtre séparée. La couleur des chandeliers se change chaques**N** ticks par hasard. Le paramètre N est porté auxparamètres extérieurs de l'indicateur pour la possibilité de l'installation manuelle (l'onglet "Paramètres" dans les propriétés de l'indicateur).

© 2000-2015, MetaQuotes Software Corp.

Prêtez attention que primordialement à la construction graphique **plot1**on spécifie la couleur à l'aide de la directive du compilateur #property, et puis dans la fonction OnCalculate() une nouvelle couleur est spécifiée par hasard de la liste d'avance préparée.

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_CANDLES.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "L'indicateur pour la démonstration DRAW_COLOR_CANDLES."
#property description "Dessine dans la fenêtre séparée par les couleurs différentes le
#property description " "
#property description "La couleur et l'épaisseur et le symbole des bougies se changent
#property description "par hasard par chaque N ticks."

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
//--- spécifions 8 couleurs pour colorer les bougies (ils se trouvent dans un tableau
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- les paramètres input
input int      N=5;              // le nombre de ticks pour le changement de l'aspect
input int      bars=500;         // combien de bougies montrer
input bool     messages=false;   // l'affichage des messages dans le journal "Experts"
//--- les tampons d'indicateur
double         ColorCandlesBuffer1[];
double         ColorCandlesBuffer2[];
double         ColorCandlesBuffer3[];
double         ColorCandlesBuffer4[];
double         ColorCandlesColors[];
int            candles_colors;
//--- le nom du symbole
string symbol;
//--- le tableau pour le stockage des couleurs contient 14 éléments
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- s'il y a trop peu de barres - terminons le travail avant terme
   if(bars<50)
     {
      Comment("Spécifiez un plus grand nombre de barres! Le travail de l'indicateur es
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorCandlesBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorCandlesBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorCandlesBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorCandlesBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- une valeur vide
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---le nom du symbole, selon lequel les barres se dessinent
   symbol=_Symbol;
//--- établissons l'affichage du symbole
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
//---- le nombre de couleurs pour colorier les bougies
   candles_colors=8;     //  à voir le commentaire pour la propriété #property indicat
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- calculons les ticks pour le changement du style et de la couleur
   ticks++;
//---si s'est accumulé le nombre suffisant des ticks
   if(ticks>=N)
     {
      //--- choisirons un nouveau symbole de la fenêtre "Aperçu du marché"
      symbol=GetRandomSymbolName();
      //--- remplaçons l'aspect
      ChangeLineAppearance();
      //--- changeons les couleurs, par lesquelles les barres sont dessinées
      ChangeColors(colors,candles_colors);

      int tries=0;
      //--- faisons 5 tentatives de remplir les tampons plot1 par les prix du symbol
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            ColorCandlesBuffer1,ColorCandlesBuffer2,ColorCandlesBuffer3,
            ColorCandlesBuffer4,ColorCandlesColors,candles_colors)
            && tries<5)
        {
         //--- le compteur des appels de la fonction CopyFromSymbolToBuffers()
         tries++;
        }
      //---oblitérons le compteur des ticks au zéro
      ticks=0;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplit une bougie indiquée                                      |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                                double &buff2[],
                                double &buff3[],
                                double &buff4[],
                                double &col_buffer[],
                                int    cndl_colors
                                )
  {
//--- copions les prix Open, High, Low et Close au tableau rates[]
   MqlRates rates[];
//--- le compteur des tentatives
   int attempts=0;
//--- combien on a été copié
   int copied=0;
//--- faisons 25 tentatives de recevoir la série temporelle selon le symbole nécessai
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- si on n'a pas réussi à copier le nombre suffisant des barres
   if(copied!=bars)
     {
      //--- formons la chaîne du message
      string comm=StringFormat("On a réussi à recevoir seulement %d barres de %d barre
                               name,
                               copied,
                               bars
                               );
      //--- déduisons le message au commentaire sur une principale fenêtre du graphiqu
      Comment(comm);
      //--- déduisons les messages
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- établissons l'affichage du symbole
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
     }
//--- initialisons les tampons par les valeurs vides
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- copions les prix aux tampons sur chaque tick
   for(int i=0;i<copied;i++)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- calculons l'index correspondant pour les tampons
      int buffer_index=total-copied+i;
      //--- inscrivons les prix aux tampons
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
      //--- spécifions la couleur de la bougie
      int color_index=i%cndl_colors;
      col_buffer[buffer_index]=color_index;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Rend accidentellement le symbole du Market Watch                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- le nombre de symboles montrés dans la fenêtre "Aperçu du marché"
   int symbols=SymbolsTotal(true);
//--- la position du symbole dans la liste - le nombre accidentel de 0 jusqu'aux symbo
   int number=MathRand()%symbols;
//--- rendons le nom du symbole selon la position indiquée
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Change la couleur des segments des bougies                       |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- le nombre de couleurs
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- spécifions une nouvelle couleur par l'image accidentelle pour chaque index de co
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- recevrons le nombre accidentel
      int number=MathRand();
      //--- recevrons l'index dans le tableau col[] comme le reste de la division enti
      int i=number%size;
      //--- installons la couleur pour chaque index comme la propriété PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  le numéro du style graphique
                          PLOT_LINE_COLOR,      //  l'identificateur de la propriété
                          plot_color_ind,       //  l'index de la couleur, où nous ins
                          cols[i]);             //   une nouvelle couleur
      //--- inscrirons les couleurs
      comm=comm+StringFormat("CandleColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
```

© 2000-2015, MetaQuotes Software Corp.

```
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Modifie l'apparence des bougies                                  |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- la ligne pour la formation de l'information sur les propriétés des bougies
   string comm="";
//--- inscrirons le nom du symbole
   comm="\r\n"+symbol+comm;
//--- déduisons l'information sur le graphique par le commentaire
   Comment(comm);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Le lien entre les propriétés de l'indicateur et les fonctions correspondantes

Chaque indicateur d'utilisateur possède la multitude de propriétés,dont la partie est obligatoire et se trouve toujours tout au début de la description. Ce sont les propriétés:

- l'indication de la fenêtre pour l'affichage de l'indicateur - indicator_separate_window ou indicator_chart_window;
- le nombre de tampons d'indicateur - indicator_buffers;
- le nombre de constructions graphiques, qui affiche l'indicateur - indicator_plots.

Mais il y a une autre partie des propriétés, que l'on peut spécifier dans les directives du préprocesseur, ainsi que dans les fonctions pour la création de l'indicateur d'utilisateur. Dans le tableau on énumère ces propriétés et les fonctions leur correspondant.

Les directives pour la propriété de la sous-fenêtre de l'indicateur	Fonctions du type IndicatorSet...()	Description de la propriété établie de sous- fenêtre
indicator_height	IndicatorSetInteger (INDICATOR_INDICATOR_HEIGHT, nHeight)	La valeur de hauteur fixe de sous- fenêtre
indicator_minimum	IndicatorSetDouble (INDICATOR_MINIMUM, dMaxValue)	La valeur minimale de l'axe vertical
indicator_maximum	IndicatorSetDouble (INDICATOR_MAXIMUM, dMinValue)	La valeur maximal de l'axe vertical
indicator_levelN	IndicatorSetDouble (INDICATOR_LEVELVALUE, N-1, nLevelValue)	La valeur de l'axe vertical pour le niveau le numéro N
il n'y a pas de directive du préprocesseur	IndicatorSetString (INDICATOR_LEVELTEXT, N-1, sLevelName)	Le nom du niveau affiché
indicator_levelcolor	IndicatorSetInteger (INDICATOR_LEVELCOLOR, N-1, nLevelColor)	La couleur pour l'affichage du niveau le numéro N
indicator_levelwidth	IndicatorSetInteger (INDICATOR_LEVELWIDTH, N-1, nLevelWidth)	L'épaisseur de la ligne pour l'affichage du niveau le numéro N
indicator_levelstyle	IndicatorSetInteger (INDICATOR_LEVELSTYLE, N-1, nLevelStyle)	Le style de la ligne pour l'affichage du niveau le numéro N
Directives pour la propriété des constructions graphiques	**Fonctions du type PlotIndexSet...()**	**Description de la propriété établie pour la construction graphique**

© 2000-2015, MetaQuotes Software Corp.

indicator_labelN	PlotIndexSetString(N-1,PLOT_LABEL,sLabel)	Le nom bref pour la construction graphique le numéro N. Il est affiché dans la fenêtre DataWindow et dans l'aide émergeant à l'induction du curseur sur la construction
indicator_colorN	PlotIndexSetInteger(N-1, PLOT_LINE_COLOR, nColor)	La couleur de la ligne pour la construction graphique le numéro N
indicator_styleN	PlotIndexSetInteger(N-1, PLOT_LINE_STYLE, nType)	Le style de la ligne pour la construction graphique le numéro N
indicator_typeN	PlotIndexSetInteger(N-1, PLOT_DRAW_TYPE, nType)	Le type de la ligne pour la construction graphique le numéro N
indicator_widthN	PlotIndexSetInteger(N-1, PLOT_LINE_WIDTH, nWidth)	L'épaisseur de la ligne pour la construction graphique le numéro N
Propriétés communes de l'indicateur	**Fonctions du type IndicatorSet...()**	**Description**
il n'y a pas de directive du préprocesseur	IndicatorSetString (INDICATOR_SHORTNAME, sShortName)	Établit le nom confortable court de l'indicateur, qui sera affiché dans le répertoire des indicateurs (s'ouvre dans le terminal en appuyant sur **Ctrl +I**).
il n'y a pas de directive du préprocesseur	IndicatorSetInteger (INDICATOR_DIGITS, nDigits)	Établit l'exactitude demandée pour l'affichage des valeurs de l'indicateur - le nombre de signes après le point décimal
il n'y a pas de directive du préprocesseur	IndicatorSetInteger (INDICATOR_LEVELS, nLevels)	Spécifie le nombre de niveaux sur la fenêtre de l'indicateur
indicator_applied_price	Les fonctions sont absentes, la propriété s'établit seulement par la directive du préprocesseur.	Le type du prix par défaut, utilisé pour le calcul des valeurs de l'indicateur. Est indiqué en cas de nécessité seulement à l'utilisation de la fonction OnCalculate() du premier type. On peut aussi spécifier la valeur de la propriété du dialogue des propriétés de l'indicateur sur l'onglet "Paramètres" - "Appliquer vers".

© 2000-2015, MetaQuotes Software Corp.

Il est nécessaire de noter, que le numérotage des niveaux et les constructions graphiques dans les termes du préprocesseur commence par l'unité, pendant que le numérotage des mêmes propriétés à l'utilisation des fonctions commence par le zéro, c'est-à-dire il faut indiquer la valeur, moins sur 1 que nous indiquerions avec l'utilisation #property.

Il y a quelques directives, pour lesquelles il n'y a pas fonctions correspondantes:

Directive	Description
indicator_chart_window	L'indicateur se montre dans une fenêtre principale
indicator_separate_window	L'indicateur se montre dans une sous-fenêtre séparée
indicator_buffers	Indique le nombre de tampons demandés d'indicateur
indicator_plots	Indique le nombre de constructions graphiques dans l'indicateur

© 2000-2015, MetaQuotes Software Corp.

SetIndexBuffer

Attache le tampon d'indicateur indiquée avec le tableau unidimentionnel dynamique du type double.

```
bool  SetIndexBuffer(
   int                   index,         // index du tampon
   double                buffer[],      // tableau
   ENUM_INDEXBUFFER_TYPE  data_type      // ce qui sera conservé
   );
```

Paramètres

index

[in] Le numéro du tampon d'indicateur. Le numérotage commence par 0. Le numéro doit être moins de la valeur déclarée à #property indicator_buffers.

buffer[]

[in] Le tableau déclaré dans le programme de l'indicateur d'utilisateur.

data_type

[in] Le type de données, qui se trouve dans le tableau d'indicateur. Par défaut INDICATOR_DATA (la valeur de l'indicateur calculé). Peut aussi accepter la valeur INDICATOR_COLOR_INDEX, alors ce tampon est destiné au stockage des index des couleurs pour le tampon précédent d'indicateur. On peut spécifier environ 64 couleurs dans la ligne #property indicator_colorN. La valeur INDICATOR_CALCULATIONS signifie que ce tampon participe aux calculs intermédiaires de l'indicateur et n'est pas destiné pour le dessin.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

Note

Après le binding le tableau dynamique *buffer[]* aura l'indexation comme dans les tableaux ordinaires, même si pour le tableau lié on établit préalablement l'indexation comme aux séries temporelles. S'il est nécessaire de changer l'ordre d'accès vers les éléments du tableau d'indicateur, il faut appliquer la fonction ArraySetAsSeries() après le binding du tableau par la fonction SetIndexBuffer(). Il faut prendre en attention qu'il est impossible de changer la grandeur pour les tableaux dynamiques, nommés comme les tampons d'indicateur par la fonction SetIndexBuffer(). Pour les tampons d'indicateur toutes les opérations du changement de la grandeur sont produites par le sous-système exécutant du terminal.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                TestCopyBuffer1.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input bool               AsSeries=true;
input int                period=15;
input ENUM_MA_METHOD     smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
input int                shift=0;
//--- indicator buffers
double                   MABuffer[];
int                      ma_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   if(AsSeries) ArraySetAsSeries(MABuffer,true);
   Print("Le tampon d'indicateur est la série temporelle = ",ArrayGetAsSeries(MABuffer
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Le tampon d'indicateur après SetIndexBuffer() est la série temporelle = ",
         ArrayGetAsSeries(MABuffer));

//--- changeons le numéro d'accès vers les éléments du tampon d'indicateur
   ArraySetAsSeries(MABuffer,AsSeries);

   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                const long &volume[],
                const int &spread[])
  {
//--- copions les valeurs du glissant moyen au tampon MABuffer
   int copied=CopyBuffer(ma_handle,0,0,rates_total,MABuffer);

   Print("MABuffer[0] = ",MABuffer[0]);// en fonction de la valeur AsSeries
                                       // recevrons la plus vieille valeur
                                       // ou sur la barre courante inachevée

//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Voir aussi

Les propriétés des indicateurs d'utilisateur, l'accès vers les séries temporelles et les indicateurs

© 2000-2015, MetaQuotes Software Corp.

IndicatorSetDouble

Spécifie la valeur de la propriété correspondante de l'indicateur. La propriété de l'indicateur doit être du type double. Il y a 2 variantes de la fonction.

L'appel avec l'indication de l'identificateur de la propriété.

```
bool  IndicatorSetDouble(
   int     prop_id,           // identificateur
   double  prop_value         // valeur établie
   );
```

L'appel avec l'indication de l'identificateur et le modificateur de la propriété.

```
bool  IndicatorSetDouble(
   int     prop_id,           // identificateur
   int     prop_modifier,     // modificateur
   double  prop_value         // valeur établie
   )
```

Paramètres

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_CUSTOMIND_PROPERTY_DOUBLE.

prop_modifier

[in] Le modificateur de la propriété indiquée. Seulement les propriétés des niveaux demandent le modificateur. Le numérotage des niveaux commence par 0, c'est-à-dire pour spécifier la propriété du deuxième niveau il faut indiquer 1 (moins 1 que lors de l'utilisation de la directive du compilateur).

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

Note

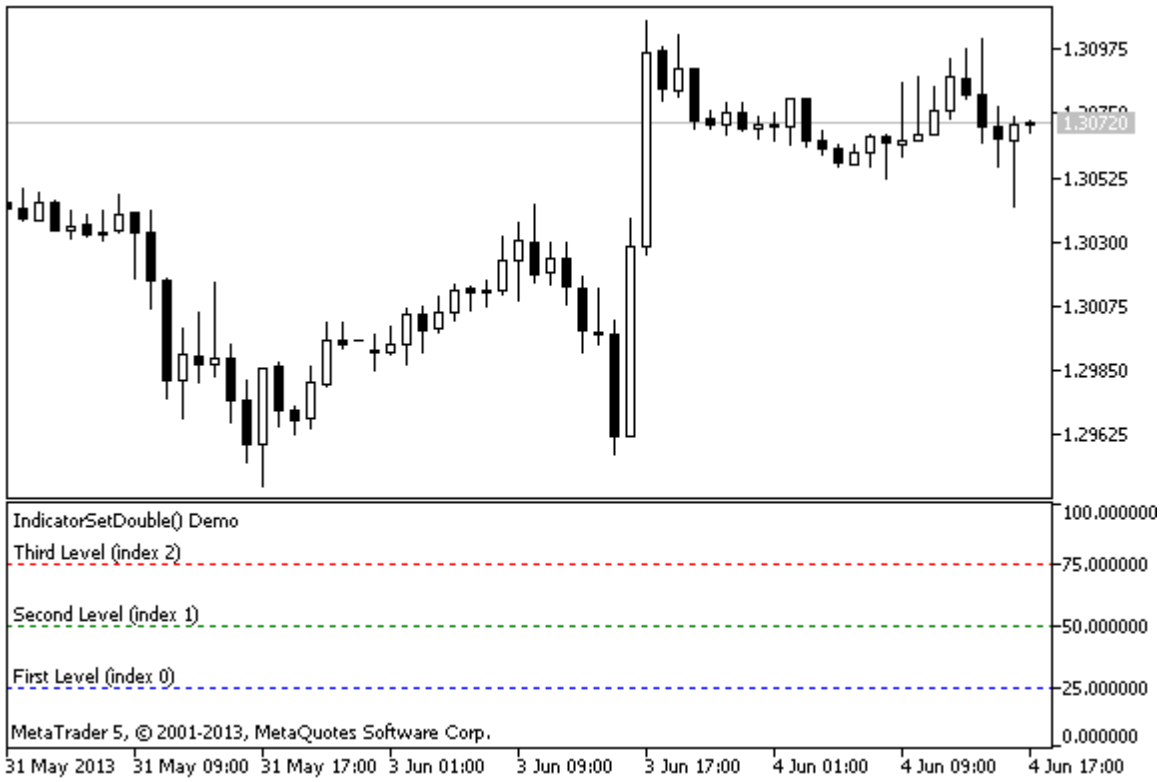
Le numérotage des propriétés (des modificateurs) à l'utilisation de la directive #property commence par 1 (un), pendant que la fonction utilise le numérotage par 0 (le zéro). Si le numéro du niveau était spécifié incorrectementl'affichage de l'indicateur peut se distinguer de celui qui est supposé.

Par exemple, on peut spécifier la valeur du premier niveau pour l'indicateur dans le sous- fenêtre séparé par deux moyens:

- property indicator_level**1** 50 - 1 est utilisé pour l'indication du numéro du niveau,
- IndicatorSetDouble(INDICATOR_LEVELVALUE, **0**, 50) - 0 est utilisé pour l'indication du premier niveau.

L'exemple: l'indicateur -"changeling", changeant les valeurs maximale et minimale de la fenêtre de l'indicateur, ainsi que les valeurs des niveaux, sur lesquels se trouvent les lignes horizontales.

© 2000-2015, MetaQuotes Software Corp.




```
#property indicator_separate_window
//--- établirons les valeurs maximale et minimale pour la fenêtre de l'indicateur
#property indicator_minimum  0
#property indicator_maximum  100
//--- spécifions la démonstration de trois niveaux horizontaux dans la fenêtre séparée
#property indicator_level1 25
#property indicator_level2 50
#property indicator_level3 75
//--- établirons l'épaisseur des niveaux horizontaux
#property indicator_levelwidth 1
//--- établirons le style des niveaux horizontaux
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- spécifions les descriptions des niveaux horizontaux
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//---spécifions le nom court de l'indicateur
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetDouble() Demo");
//--- spécifions à chaque niveau son couleur
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,0,clrBlue);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,1,clrGreen);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,2,clrRed);
```

© 2000-2015, MetaQuotes Software Corp.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
   static double level1=25,level2=50,level3=75;
   static double max=100,min=0, shift=100;
//--- comptons les ticks
   tick_counter++;
//--- faisons un coup sur chaque 10-ème tick
   if(tick_counter%10==0)
     {
      //--- retournerons le signe pour les valeurs du niveau
      level1=-level1;
      level2=-level2;
      level3=-level3;
      //--- retournerons le signe pour les valeurs maximales et minimales
      max-=shift;
      min-=shift;
      //--- retournerons la valeur du décalage
      shift=-shift;
      //--- spécifions les nouvelles valeurs des niveaux
      IndicatorSetDouble(INDICATOR_LEVELVALUE,0,level1);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,1,level2);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,2,level3);
      //--- spécifions les nouvelles valeurs maximales et minimales pour la fenêtre de
      Print("Set up max = ",max,",   min = ",min);
      IndicatorSetDouble(INDICATOR_MAXIMUM,max);
      IndicatorSetDouble(INDICATOR_MINIMUM,min);
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Styles des indicateurs dans les exemples, Lien entre les propriétés de l'indicateur et les fonctions

© 2000-2015, MetaQuotes Software Corp.

correspondantes, Styles du dessin

© 2000-2015, MetaQuotes Software Corp.

IndicatorSetInteger

Spécifie la valeur de la propriété correspondante de l'indicateur. La propriété de l'indicateur doit être du type int ou color. Il y a 2 variantes de la fonction.

L'appel avec l'indication de l'identificateur de la propriété.

```
bool  IndicatorSetInteger(
   int  prop_id,           // identificateur
   int  prop_value         // valeur établie
   );
```

L'appel avec l'indication de l'identificateur et le modificateur de la propriété.

```
bool  IndicatorSetInteger(
   int  prop_id,           // identificateur
   int  prop_modifier,     // modificateur
   int  prop_value         // valeur établie
   )
```

Paramètres

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_CUSTOMIND_PROPERTY_INTEGER.

prop_modifier

[in] Le modificateur de la propriété indiquée. Seulement les propriétés des niveaux demandent le modificateur.

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

Note

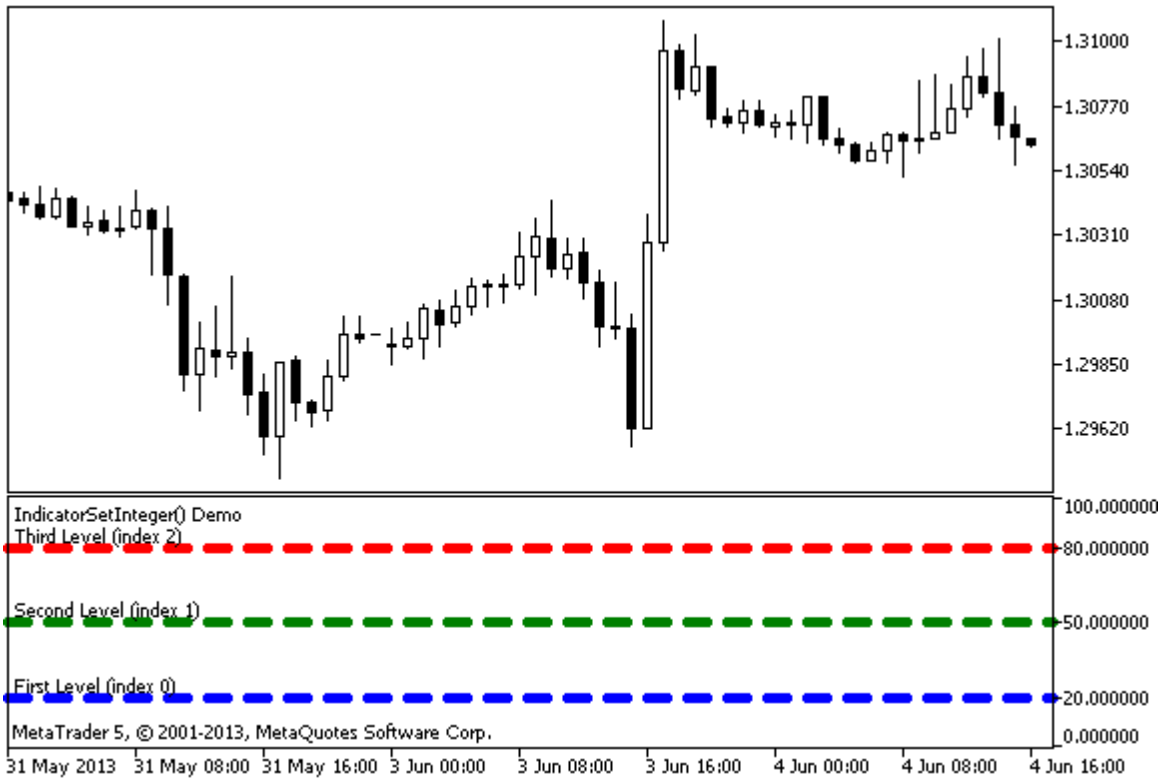
Le numérotage des propriétés (des modificateurs) à l'utilisation de la directive #property commence par 1 (une unité), pendant que la fonction utilise le numérotage par 0 (un zéro). Si le numéro du niveau était spécifié incorrectementl'affichage de l'indicateur peut se distinguer de celui qui est supposé.

Par exemple pour spécifier l'épaisseur de la ligne du premier niveau horizontal utilisez l'indice nul:

- IndicatorSetInteger(INDICATOR_LEVELWIDTH, **0**, 5) - on utilise l'indice 0 pour spécifier l'épaisseur de la ligne du premier niveau.

L'exemple: l'indicateur qui spécifie la couleur, le style et l'épaisseur des niveaux horizontaux de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.



© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- spécifions la démonstration de trois niveaux horizontaux dans la fenêtre séparée
#property indicator_level1 20
#property indicator_level2 50
#property indicator_level3 80
//---établissons l'épaisseur des niveaux horizontaux
#property indicator_levelwidth 5
//--- établissons la couleur des niveaux horizontaux
#property indicator_levelcolor clrAliceBlue
//--- établissons le style des niveaux horizontaux
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- spécifions les descriptions des niveaux horizontaux
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- spécifions le nom court de l'indicateur
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetInteger() Demo");
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
//--- calculons les ticks
   tick_counter++;
//--- établissons les couleurs des niveaux horizontaux en fonction du compteur des tic
   ChangeLevelColor(0,tick_counter,3,6,10); // trois derniers paramètres basculent la
   ChangeLevelColor(1,tick_counter,3,6,8);
   ChangeLevelColor(2,tick_counter,4,7,9);
//--- changeons les styles des niveaux horizontaux
   ChangeLevelStyle(0,tick_counter);
   ChangeLevelStyle(1,tick_counter+5);
   ChangeLevelStyle(2,tick_counter+15);
//--- recevrons l'épaisseur de la ligne, comme le reste de la division entière du nomb
   int width=tick_counter%5;
//--- passerons selon tous les niveaux horizontaux et exposerons
   for(int l=0;l<3;l++)
      IndicatorSetInteger(INDICATOR_LEVELWIDTH,l,width+1);
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| établissons la couleur de la ligne horizontale dans la fenêtre séparée de l'indica
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
void ChangeLevelColor(int level,      // le numéro de la ligne horizontale
                      int tick_number,// le dividende, le nombre pour la réception du
                      int f_trigger,  // le premier diviseur du basculement de la coul
                      int s_trigger,  // le deuxième diviseur du basculement de la cou
                      int t_trigger)  // le troisième diviseur du basculement de la co
  {
   static color colors[3]={clrRed,clrBlue,clrGreen};
//--- l'indice de la couleur du tableau colors[]
   int index=-1;
//--- calculons le numéro de la couleur du tableau colors [] pour colorier la ligne ho
   if(tick_number%f_trigger==0)
      index=0;   // si le nombre tick_number se divise sans reste sur f_trigger
   if(tick_number%s_trigger==0)
      index=1;   // si le nombre tick_number se divise sans reste sur s_trigger
   if(tick_number%t_trigger==0)
      index=2;   // si le nombre tick_number se divise sans reste sur t_trigger
//--- si la couleur est définie, l'établissons
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELCOLOR,level,colors[index]);
//---
  }
//+------------------------------------------------------------------+
//| établissons le style de la ligne horizontale dans la fenêtre séparée de l'indicate
//+------------------------------------------------------------------+
void ChangeLevelStyle(int level,      // le numéro de la ligne horizontale
                      int tick_number// le nombre pour la réception du reste de la div
                      )
  {
//--- le tableau pour stocker les styles
   static ENUM_LINE_STYLE styles[5]=
     {STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//--- l'indice du style du tableau styles[]
   int index=-1;
//--- calculons le numéro du tableau styles[] pour établir le style de la ligne horizo
   if(tick_number%50==0)
      index=5;   // si tick_number se divise sans reste sur 50, le style STYLE_DASHDOT
   if(tick_number%40==0)
      index=4;   // ... le style STYLE_DASHDOT
   if(tick_number%30==0)
      index=3;   // ... STYLE_DOT
   if(tick_number%20==0)
      index=2;   // ... STYLE_DASH
   if(tick_number%10==0)
      index=1;   // ... STYLE_SOLID
//--- si le style est défini, l'établissons
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELSTYLE,level,styles[index]);
  }
```

Voir aussi

Propriétés des indicateurs d'utilisateurs, Propriétés des programmes (#property), Styles du dessin

© 2000-2015, MetaQuotes Software Corp.

IndicatorSetString

Spécifie la valeur de la propriété correspondante de l'indicateur. La propriété de l'indicateur doit être du type string. Il y a 2 variantes de la fonction.

L'appel avec l'indication de l'identificateur de la propriété.

```
bool  IndicatorSetString(
   int     prop_id,           // identificateur
   string  prop_value         // valeur établie
   );
```

L'appel avec l'indication de l'identificateur et le modificateur de la propriété.

```
bool  IndicatorSetString(
   int     prop_id,           // identificateur
   int     prop_modifier,     // modificateur
   string  prop_value         // valeur établie
   )
```

Paramètres

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_CUSTOMIND_PROPERTY_STRING.

prop_modifier

[in] Le modificateur de la propriété indiquée. Seulement les propriétés des niveaux demandent le modificateur.

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

Note

Le numérotage des propriétés (des modificateurs) à l'utilisation de la directive #property commence par 1 (une unité), pendant que la fonction utilise le numérotage par 0 (un zéro). Si le numéro du niveau était spécifié incorrectementl'affichage de l'indicateur peut se distinguer de celui qui est supposé.

Par exemple pour spécifier la description du premier niveau horizontal utilisez l'indice nul:

- IndicatorSetString(INDICATOR_LEVELTEXT, **0**, "First Level") - on utilise l'indice 0 pour spécifier la tâche de la description de texte du premier niveau.

L'exemple: l'indicateur qui établie les signatures vers les niveaux horizontaux de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.


```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- spécifions la démonstration de trois niveaux horizontaux dans la fenêtre séparée
#property indicator_level1 30
#property indicator_level2 50
#property indicator_level3 70
//---établissons la couleur des niveaux horizontaux
#property indicator_levelcolor clrRed
//--- établissons le style des niveaux horizontaux
#property indicator_levelstyle STYLE_SOLID
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- spécifions les descriptions des niveaux horizontaux
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//---spécifions le nom court de l'indicateur
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetString() Demo");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---

//--- return value of prev_calculated for next call
   return(rates_total);
  }
```

Voir aussi

Propriétés des indicateurs d'utilisateurs, Propriétés des programmes (#property)

© 2000-2015, MetaQuotes Software Corp.

PlotIndexSetDouble

Spécifie la valeur de la propriété correspondante de la ligne correspondante de l'indicateur. La propriété de l'indicateur doit être du type double.

```
bool  PlotIndexSetDouble(
   int     plot_index,     // index du style graphique
   int     prop_id,        // identificateur de la propriété
   double  prop_value      // valeur établie
   );
```

Paramètres

plot_index

[in] L'index de la construction graphique

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_PLOT_PROPERTY_DOUBLE.

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

© 2000-2015, MetaQuotes Software Corp.

PlotIndexSetInteger

Spécifie la valeur de la propriété correspondante de la ligne correspondante de l'indicateur. La propriété de l'indicateur doit être du typeint, char, bool ou color. Il y a 2 variantes de la fonction.

L'appel avec l'indication de l'identificateur de la propriété.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // index du style graphique
   int  prop_id,            // identificateur de la propriété
   int  prop_value          // valeur établie
   );
```

L'appel avec l'indication de l'identificateur et le modificateur de la propriété.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // index du style graphique
   int  prop_id,            // identificateur de la propriété
   int  prop_modifier,      // modificateur de la propriété
   int  prop_value          // valeur établie
   )
```

Paramètres

plot_index

[in] L'index de la construction graphique

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Le modificateur de la propriété indiquée. Seulement les propriétés des index des couleurs demandent le modificateur.

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

Exemple: l'indicateur qui dessine la ligne tricolore. Le schéma coloré change dans chaque 5 ticks.

© 2000-2015, MetaQuotes Software Corp.


```
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//---- plot ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrGreen,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- indicator buffers
double         ColorLineBuffer[];
double         ColorBuffer[];
int            MA_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBuffer,INDICATOR_COLOR_INDEX);
//--- get MA handle
   MA_handle=iMA(Symbol(),0,10,0,MODE_EMA,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| get color index                                                  |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
int getIndexOfColor(int i)
  {
   int j=i%300;
   if(j<100) return(0);// first index
   if(j<200) return(1);// second index
   return(2); // third index
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   static int ticks=0,modified=0;
   int limit;
//--- first calculation or number of bars was changed
   if(prev_calculated==0)
     {
      //--- copy values of MA into indicator buffer ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);// copying failed - throw away
      //--- now set line color for every bar
      for(int i=0;i<rates_total;i++)
         ColorBuffer[i]=getIndexOfColor(i);
     }
   else
     {
      //--- copy values of MA into indicator buffer ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);

      ticks++;// ticks counting
      if(ticks>=5)//it's time to change color scheme
        {
         ticks=0; // reset counter
         modified++; // counter of color changes
         if(modified>=3)modified=0;// reset counter
         ResetLastError();
         switch(modified)
           {
```

© 2000-2015, MetaQuotes Software Corp.

```
            case 0:// first color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrRed);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrBlue);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrGreen);
               Print("Color scheme "+modified);
               break;
            case 1:// second color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrYellow);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrPink);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLightSlateGray);
               Print("Color scheme "+modified);
               break;
            default:// third color scheme
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrLightGoldenrod);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrOrchid);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLimeGreen);
               Print("Color scheme "+modified);
           }
        }
      else
        {
         //--- set start position
         limit=prev_calculated-1;
         //--- now we set line color for every bar
         for(int i=limit;i<rates_total;i++)
            ColorBuffer[i]=getIndexOfColor(i);
        }
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

PlotIndexSetString

Spécifie la valeur de la propriété correspondante de la ligne correspondante de l'indicateur. La propriété de l'indicateur doit être du type string.

```
bool  PlotIndexSetString(
   int     plot_index,     // index du style graphique
   int     prop_id,        // identificateur de la propriété
   string  prop_value      // valeur établie
   );
```

Paramètres

plot_index

[in] L'index de la construction graphique

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_PLOT_PROPERTY_STRING.

prop_value

[in] La valeur de la propriété.

La valeur rendue

En cas de l'exécution réussie rend true, dans le cas contraire rend false.

© 2000-2015, MetaQuotes Software Corp.

PlotIndexGetInteger

Spécifie la valeur de la propriété correspondante de la ligne correspondante de l'indicateur. La propriété de l'indicateur doit être du type int, color, bool ou char. Il y a 2 variantes de la fonction.

L'appel avec l'indication de l'identificateur de la propriété.

```
int  PlotIndexGetInteger(
   int  plot_index,         // index du style graphique
   int  prop_id,            // identificateur de la propriété
   );
```

L'appel avec l'indication de l'identificateur et le modificateur de la propriété.

```
int  PlotIndexGetInteger(
   int  plot_index,         // index du style graphique
   int  prop_id,            // identificateur de la propriété
   int  prop_modifier       // identificateur de la propriété
   )
```

Paramètres

plot_index

[in] L'index de la construction graphique

prop_id

[in] L'identificateur de la propriété de l'indicateur. La valeur peut être une des valeurs de l'énumération ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Le modificateur de la propriété indiquée. Seulement les propriétés des index des couleurs demandent le modificateur.

Note

La fonction est destinée à l'extrait des ajustements du dessin de la ligne correspondante de l'indicateur. La fonction travaille avec la fonction PlotIndexSetInteger pour le copiage des propriétés du dessin d'une ligne à l'autre.

Exemple: l'indicateur qui colorie des bougies selon le jour de la semaine. Les couleurs pour chaque jour sont spécifiés d'une manière de programme.

© 2000-2015, MetaQuotes Software Corp.


```
#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- indicator buffers
double          OpenBuffer[];
double          HighBuffer[];
double          LowBuffer[];
double          CloseBuffer[];
double          ColorCandlesColors[];
color           ColorOfDay[6]={CLR_NONE,clrMediumSlateBlue,
                               clrDarkGoldenrod,clrForestGreen,clrBlueViolet,clrRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,OpenBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,HighBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LowBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,CloseBuffer,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- set number of colors in color buffer
```

© 2000-2015, MetaQuotes Software Corp.

```
   PlotIndexSetInteger(0,PLOT_COLOR_INDEXES,6);
//--- set colors for color buffer
   for(int i=1;i<6;i++)
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,i,ColorOfDay[i]);
//--- set accuracy
   IndicatorSetInteger(INDICATOR_DIGITS,_Digits);
   printf("We have %u colors of days",PlotIndexGetInteger(0,PLOT_COLOR_INDEXES));
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   int i;
   MqlDateTime t;
//----
   if(prev_calculated==0) i=0;
   else i=prev_calculated-1;
//----
   while(i<rates_total)
     {
      OpenBuffer[i]=open[i];
      HighBuffer[i]=high[i];
      LowBuffer[i]=low[i];
      CloseBuffer[i]=close[i];
      //--- set color for every candle
      TimeToStruct(time[i],t);
      ColorCandlesColors[i]=t.day_of_week;
      //---
      i++;
     }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

Les objets graphiques

Le groupe des fonctions destinées au travail avec les objets graphiques, se rapportant à n'importe quel graphique indiqué. On ne peut pas utiliser ces fonctions dans les indicateurs.

Les fonctions qui définissent les propriétés des objets graphiques, ainsi que les opérations de la création de ObjectCreate() et de déplacement ObjectMove() des objets sur le graphique, servent en réalité à envoyer des commandes au graphique. À l'exécution réussie de ces fonctions la commande se trouve dans une file d'attente des événements du graphique. Le changement visuel des propriétés des objets graphiques s'est produit en train du traitement de la file d'attente des événements du graphique donné.

Pour cette raison il ne faut pas attendre la mise à jour visuelle immédiate des objets graphiques après l'appel des fonctions données. En général, la mise à jour des objets graphiques s'est produite par le terminal automatiquement selon les événements du changement - l'entrée de la nouvelle cotation, le changement de la taille de la fenêtre du graphique etc.

Pour la mise à jour forcée des objets graphiques utilisez la commande pour redessiner le graphique ChartRedraw().

Fonction	Action
ObjectCreate	Crée l'objet du type spécifié sur le graphique indiqué
ObjectName	Rend le nom de l'objet du type correspondant dans le graphique indiqué (indiqué dans la sous-fenêtre du graphique)
ObjectDelete	Supprime l'objet avec le nom indiqué du graphique indiqué (de la sous-fenêtre de graphique indiquée)
ObjectsDeleteAll	Supprime tous les objets du type indiqué du graphique indiqué (de la sous-fenêtre de graphique indiquée)
ObjectFind	Cherche l'objet selon le nom avec l'identificateur indiqué
ObjectGetTimeByValue	Rend la valeur du temps pour la valeur indiquée du prix de l'objet
ObjectGetValueByTime	Rend la valeur de prix de l'objet pour le temps indiqué
ObjectMove	Change les coordonnées du point indiqué du rattachement de l'objet
ObjectsTotal	Rend le nombre d'objets du type indiqué dans le graphique indiqué (la sous-fenêtre de graphique indiquée)
ObjectGetDouble	Rend la valeur du type double de la propriété correspondante de l'objet

© 2000-2015, MetaQuotes Software Corp.

ObjectGetInteger	Rend la valeur entière de la propriété correspondante de l'objet
ObjectGetString	Rend la valeur du type string de la propriété correspondante de l'objet
ObjectSetDouble	Met la valeur de la propriété correspondante de l'objet
ObjectSetInteger	Met la valeur de la propriété correspondante de l'objet
ObjectSetString	Met la valeur de la propriété correspondante de l'objet
TextSetFont	Établit la fonte pour la sortie du texte par les méthodes du dessin (on utilise par défaut la fonte Arial 20)
TextOut	Déduit le texte au tableau d'utilisateur (le tampon) destiné à la création de la ressourcegraphique
TextGetSize	Rend la largeur et la hauteur de la ligne aux réglages courants de la fonte

Chaque objet graphique doit avoir le nom, unique dans la limite d'un graphique, y compris ses sous-fenêtres. Le changement du nom de l'objet graphique forme deux événements: premier est un événement de l'éloignement de l'objet avec un vieux nom, et deuxième – l'événement de la création de l'objet graphique avec un nouveau nom.

Après la création de l'objet ou la modification des propriétés de l'objet il est recommandé d'appeler la fonction ChartRedraw(), qui rend au terminal l'ordre pour le dessin forcé du graphique (et de tous les objets visibles sur lui).

© 2000-2015, MetaQuotes Software Corp.

ObjectCreate

Crée l'objet avec le nom indiqué, le type et les coordonnées initiales à la sous-fenêtre indiquée du graphique. On peut indiquer environ 30 coordonnées pendant la création.

```
bool  ObjectCreate(
   long          chart_id,      // identificateur du graphique
   string        name,          // nom de l'objet
   ENUM_OBJECT   type,          // type de l'objet
   int           sub_window,    // index de la fenêtre
   datetime      time1,         // temps du premier point du rattachement
   double        price1,        // prix du premier point du rattachement
   ...
   datetime      timeN=0,       // temps du N-ème point du rattachement
   double        priceN=0,      // prix du N-ème point du rattachement
   ...
   datetime      time30=0,      // temps du 30-ème point du rattachement
   double        price30=0      // prix du 30-ème point du rattachement
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet. Le nom doit être unique dans la limite d'un graphique, y compris ses sous-fenêtres.

type

[in] Le type de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT.

sub_window

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie une fenêtre principale du graphique. La sous-fenêtre indiquée doit exister, dans le cas contraire la fonction rend false.

time1

[in] La coordonnée temporaire du premier rattachement.

price1

[in] La coordonnée de prix du premier point du rattachement.

timeN=0

[in] La coordonnée temporaire du N-ème point du rattachement.

priceN=0

[in] La coordonnée de prix du N-ème point du rattachement.

time30=0

[in] La coordonnée temporaire du trentième point du rattachement.

price30=0

© 2000-2015, MetaQuotes Software Corp.

[in] La coordonnée de prix du trentième point du rattachement.

La valeur rendue

Rend true ou false en fonction de celui-là est-ce que l'objet a été créé ou non. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError(). Si l'objet était créé jusqu'à cela, on produit la tentative de changer ses coordonnées.

Note

Имя графического объекта не должно превышать 63 символа.

Le numérotage des sous-fenêtres du graphique (si sur le graphique il y a les sous-fenêtres avec les indicateurs) commence par 1. Une principale fenêtre du graphique existe toujours et elle a l'index 0.

Une grande quantité de points du rattachement (jusqu'à 30) est prévue pour la future utilisation. En même temps la limitations seulement par 30 points possibles du rattachement pour les objets graphiques est conditionnée par ce qu'à l'appel de la fonction le nombre de paramètres ne doit pas excéder 64.

Au changement de nom de l'objet graphique on forme simultanément deux événements, lesquels on peut traiter dans l'expert ou l'indicateur par la fonction OnChartEvent():

- l'événement de l'effacement de l'objet avec un vieux nom;
- l'événement de la création de l'objet graphique avec un nouveau nom.

Pour créer chacun destypes des objetsil faut spécifier la quantité définie de points du rattachement:

Identificateur	La description	Points du rattachement
OBJ_VLINE	La ligne verticale	Un point du rattachement. En fait, on utilise seulement la coordonnée selon l'axe du temps.
OBJ_HLINE	Ligne horizontale	Un point du rattachement. En fait, on utilise seulement la coordonnée selon l'axe du prix.
OBJ_TREND	La ligne de tendance	Deux points du rattachement
OBJ_TRENDBYANGLE	La ligne de tendance selon l'angle	Deux points du rattachement
OBJ_CYCLES	Lignes cycliques	Deux points du rattachement
OBJ_ARROWED_LINE	Objet "La ligne avec la flèche"	Deux points du rattachement
OBJ_CHANNEL	Le canal équidistant	Trois points du rattachement
OBJ_STDDEVCHANNEL	Le canal de la divergence standard	Deux points du rattachement
OBJ_REGRESSION	Le canal sur la régression linéaire	Deux points du rattachement

© 2000-2015, MetaQuotes Software Corp.

OBJ_PITCHFORK	Andrews' Pitchfork	Trois points du rattachement
OBJ_GANNLINE	La ligne de Gann	Deux points du rattachement
OBJ_GANNFAN	L'évantail de Gann	Deux points du rattachement
OBJ_GANNGRID	Grille de Gann	Deux points du rattachement
OBJ_FIBO	Les niveaux de Fibonacci	Deux points du rattachement
OBJ_FIBOTIMES	Les zones temporaires de Fibonacci	Deux points du rattachement
OBJ_FIBOFAN	Le fanion de Fibonacci	Deux points du rattachement
OBJ_FIBOARC	Les arcs de Fibonacci	Deux points du rattachement
OBJ_FIBOCHANNEL	Le canal de Fibonacci	Trois points du rattachement
OBJ_EXPANSION	L'extension de Fibonacci	Trois points du rattachement
OBJ_ELLIOTWAVE5	La vague d'impulsion	Cinq points du rattachement.
OBJ_ELLIOTWAVE3	La vague de correction	Trois points du rattachement
OBJ_RECTANGLE	Rectangle	Deux points du rattachement
OBJ_TRIANGLE	Triangle	Trois points du rattachement
OBJ_ELLIPSE	Ellipse	Trois points du rattachement
OBJ_ARROW_THUMB_UP	Le signe "bien" (le pouce en haut)	Un point du rattachement
OBJ_ARROW_THUMB_DOWN	Le signe "mal" (le pouce en bas)	Un point du rattachement
OBJ_ARROW_UP	Le signe "La flèche en haut"	Un point du rattachement
OBJ_ARROW_DOWN	Le signe "La flèche en haut"	Un point du rattachement
OBJ_ARROW_STOP	Le signe "Stop"	Un point du rattachement
OBJ_ARROW_CHECK	Le signe "Coche" (encoche)	Un point du rattachement
OBJ_ARROW_LEFT_PRICE	Une marque gauche de prix	Un point du rattachement
OBJ_ARROW_RIGHT_PRICE	Une marque droite de prix	Un point du rattachement
OBJ_ARROW_BUY	Le signe "Buy"	Un point du rattachement
OBJ_ARROW_SELL	Le signe "Sell"	Un point du rattachement
OBJ_ARROW	L'objet "La flèche"	Un point du rattachement
OBJ_TEXT	L'objet "Texte"	Un point du rattachement
OBJ_LABEL	L'objet "La marque de texte"	La position est définie par des propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.
OBJ_BUTTON	L'objet "Le bouton"	La position est définie par

© 2000-2015, MetaQuotes Software Corp.

		des propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.
OBJ_CHART	L'objet "Graphique"	La position est définie par des propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.
OBJ_BITMAP	L'objet "Dessin"	Un point du rattachement
OBJ_BITMAP_LABEL	L'objet "La marque graphique"	La position est définie par des propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.
OBJ_EDIT	L'objet "Champ d'entrée"	La position est définie par des propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.
OBJ_EVENT	L'objet "Événement", correspondant à l'événement dans le calendrier économique	Un point du rattachement. En fait, on utilise seulement la coordonnée selon l'axe du temps.
OBJ_RECTANGLE_LABEL	L'objet "Marque rectangulaire" pour la création et la présentation de l'interface d'utilisateur graphique.	La position est définie par des propriétésOBJPROP_XDISTANCE et OBJPROP_YDISTANCE.

© 2000-2015, MetaQuotes Software Corp.

ObjectName

Rend le nom de l'objet correspondant au chart indiqué, dans la sous- fenêtre indiquée de chart indiqué, du type indiqué.

```
string  ObjectName(
   long  chart_id,           // identificateur du graphique
   int   pos,                // numéro dans la liste des objets
   int   sub_window=-1,      // numéro de la fenêtre
   int   type=-1             // type de l'objet
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

pos

[in] Le numéro consécutif de l'objet selon le filtre indiqué selon le numéro de la sous- fenêtre et du type.

sub_window=-1

[in] Le numéro de la sous- fenêtre du graphique. 0 signifie une fenêtre principale du graphique, -1 signifie tous les sous- fenêtres du graphique, y compris une fenêtre principale.

type=-1

[in] Le type de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT. -1 signifie tous les types.

La valeur rendue

Le nom de l'objet en cas du succès.

Note

Au changement de nom de l'objet graphique on forme simultanément deux événements, lesquels on peut traiter dans l'expert ou l'indicateur par la fonction OnChartEvent():

- l'événement de l'effacement de l'objet avec un vieux nom;
- l'événement de la création de l'objet graphique avec un nouveau nom.

© 2000-2015, MetaQuotes Software Corp.

ObjectDelete

Supprime l'objet avec le nom indiqué du graphique indiqué.

```
bool  ObjectDelete(
   long    chart_id,      // identificateur du graphique
   string  name           // nom de l'objet
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet supprimé.

La valeur rendue

Rend true à l'achèvement réussi de l'opération de l'effacement, dans le cas contraire rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Au changement de nom de l'objet graphique on forme simultanément deux événements, lesquels on peut traiter dans l'expert ou l'indicateur par la fonction OnChartEvent():

- l'événement de l'effacement de l'objet avec un vieux nom;
- l'événement de la création de l'objet graphique avec un nouveau nom.

© 2000-2015, MetaQuotes Software Corp.

ObjectsDeleteAll

Supprime tous les objets au graphique indiqué, dans la sous- fenêtre indiquée du graphique indiqué, du type indiqué.

```
int  ObjectsDeleteAll(
   long  chart_id,           // identificateur du graphique
   int   sub_window=-1,      // index de la fenêtre
   int   type=-1             // type de l'objet pour l'effacement
   );
```

Removes all objects of the specified type using prefix in object names.

```
int  ObjectsDeleteAll(
   long           chart_id,  // chart ID
   const string     prefix,  // prefix in object name
   int        sub_window=-1,  // window index
   int       object_type=-1   // object type
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

prefix

[in] Prefix in object names. All objects whose names start with this set of characters will be removed from chart. You can specify prefix as 'name' or 'name*' - both variants will work the same. If an empty string is specified as the prefix, objects with all possible names will be removed.

sub_window=-1

[in] Le numéro de la sous- fenêtre du graphique. 0 signifie une fenêtre principale du graphique, -1 signifie tous les sous- fenêtres du graphique, y compris une fenêtre principale.

type=-1

[in] Le type de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT. -1 signifie tous les types.

La valeur rendue

Rend le nombre d'objets supprimés. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

ObjectFind

Cherche l'objet selon le nom avec l'identificateur indiqué.

```
int  ObjectFind(
   long    chart_id,      // identificateur du graphique
   string  name           // nom de l'objet
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet cherché.

La valeur rendue

En cas du succès la fonction rend le numéro de la sous- fenêtre (0 signifie une principale fenêtre du graphique), dans laquelle il y a un objet trouvé. Si l'objet n'est pas trouvé, la fonction rend le nombre négatif. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Au changement de nom de l'objet graphique on forme simultanément deux événements, lesquels on peut traiter dans l'expert ou l'indicateur par la fonction OnChartEvent():

- l'événement de l'effacement de l'objet avec un vieux nom;
- l'événement de la création de l'objet graphique avec un nouveau nom.

© 2000-2015, MetaQuotes Software Corp.

ObjectGetTimeByValue

Rend la valeur du temps pour la valeur indiquée du prix de l'objet indiqué.

```
datetime  ObjectGetTimeByValue(
   long    chart_id,      // identificateur du graphique
   string  name,          // nom de l'objet
   double  value,         // prix
   int     line_id        // ligne
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

value

[in] La valeur du prix.

line_id

[in] L'identificateur de la ligne.

La valeur rendue

La valeur du temps pour la valeur indiquée du prix de l'objet indiqué.

Note

Puisque l'objet dans une coordonnée du prix peut avoir quelques valeurs, il est nécessaire d'indiquer le numéro de la ligne. Cette fonction est employée seulement pour les objets suivants:

- La ligne de la tendance (OBJ_TREND)
- La ligne de la tendance selon l'angle (OBJ_TRENDBYANGLE)
- Le ligne de Gann (OBJ_GANNLINE)
- Le canal équidistant (OBJ_CHANNEL) - 2 lignes
- Le canal sur la régression linéaire (OBJ_REGRESSION) - 3 lignes
- Le canal de l'écart typique (OBJ_STDDEVCHANNEL) - 3 lignes
- La ligne avec la flèche (OBJ_ARROWED_LINE)

Voir aussi

Les types des objets

© 2000-2015, MetaQuotes Software Corp.

ObjectGetValueByTime

Rend la valeur du prix pour le temps indiqué de l'objet indiqué.

```
double  ObjectGetValueByTime(
   long      chart_id,     // identificateur du graphique
   string    name,         // nom de l'objet
   datetime  time,         // temps
   int       line_id       // ligne
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

time

[in] La valeur du temps.

line_id

[in] L'identificateur de la ligne.

La valeur rendue

La valeur du prix pour le temps indiqué de l'objet indiqué.

Note

Puisque l'objet dans une coordonnée du prix peut avoir quelques valeurs, il est nécessaire d'indiquer le numéro de la ligne. Cette fonction est employée seulement pour les objets suivants:

- La ligne de la tendance (OBJ_TREND)
- La ligne de la tendance selon l'angle (OBJ_TRENDBYANGLE)
- Le ligne de Gann (OBJ_GANNLINE)
- Le canal équidistant (OBJ_CHANNEL) - 2 lignes
- Le canal sur la régression linéaire (OBJ_REGRESSION) - 3 lignes
- Le canal de l'écart typique (OBJ_STDDEVCHANNEL) - 3 lignes
- La ligne avec la flèche (OBJ_ARROWED_LINE)

Voir aussi

Les types des objets

© 2000-2015, MetaQuotes Software Corp.

ObjectMove

Change les coordonnées du point indiqué du rattachement de l'objet.

```
bool  ObjectMove(
   long      chart_id,        // identificateur du graphique
   string    name,            // nom de l'objet
   int       point_index,     // numéro du rattachement
   datetime  time,            // temps
   double    price            // prix
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

point_index

[in] Le numéro du point du rattachement. Le nombre de points du rattachement dépend du type de l'objet.

time

[in] La coordonnée temporaire du point indiqué du rattachement.

price

[in] La coordonnée de prix du point indiqué du rattachement.

La valeur rendue

Dans le cas du succès rend true, à l'échec rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

ObjectsTotal

Rend le nom de l'objet correspondant au chart indiqué, dans la sous- fenêtre indiquée de chart indiqué, du type indiqué.

```
int  ObjectsTotal(
   long  chart_id,           // identificateur du graphique
   int   sub_window=-1,      // index de la fenêtre
   int   type=-1             // type de l'objet
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

sub_window=-1

[in] Le numéro de la sous-fenêtre du graphique. 0 signifie une fenêtre principale du graphique, -1 signifit tous les sous-fenêtres du graphique, y compris une fenêtre principale.

type=-1

[in] Le type de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT. -1 signifit tous les types.

La valeur rendue

Le nombre d'objets.

© 2000-2015, MetaQuotes Software Corp.

ObjectSetDouble

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type double. Il y a 2 variantes de la fonction.

Etablissement de la valeur de la propriété n'ayant pas le modificateur

```
bool  ObjectSetDouble(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom
   int      prop_id,           // propriété
   double   prop_value         // valeur
   );
```

Etablissement de la valeur de la propriété avec l'indication du modificateur

```
bool  ObjectSetDouble(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom
   int      prop_id,           // propriété
   int      prop_modifier,     // modificateur
   double   prop_value         // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Le modificateur de la propriété indiquée. La plupart des propriétés ne demandent pas le modificateur. Signifie le numéro du niveau dans les instruments de Fibonacci et dans l'objet graphique Andrews' Pitchfork. Le numérotage des niveux commence par le zéro.

prop_value

[in] La valeur de la propriété.

La valeur rendue

Rend true seulement dans le cas si la commande sur le changement des propriétés de l'objet graphique est envoyé avec succès au graphique, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

L'exemple de la création de l'objet Fibonacci et y ajouter un nouveau niveau

```
//| Script program start function                                    |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tableaux auxiliaires
   double high[],low[],price1,price2;
   datetime time[],time1,time2;
//---copions les prix de l'ouverture  - les 100 dernières barres sont assez
   int copied=CopyHigh(Symbol(),0,0,100,high);
   if(copied<=0)
     {
      Print("on n'a pas réussi à copier les valeurs de la série de prix High");
      return;
     }
//--- copions les prix de la clôture - les 100 dernières barres sont assez
   copied=CopyLow(Symbol(),0,0,100,low);
   if(copied<=0)
     {
      Print("On n'a pas réussi à copier les valeurs de la série de prix Low");
      return;
     }
//--- copions le temps de l'ouverture pour les 100 dernières barres
   copied=CopyTime(Symbol(),0,0,100,time);
   if(copied<=0)
     {
      Print("On n'a pas réussi à copier les valeurs de la série de prix Time");
      return;
     }
//--- organiserons l'accès aux données copiées comme aux séries temporelles - à l'enve
   ArraySetAsSeries(high,true);
   ArraySetAsSeries(low,true);
   ArraySetAsSeries(time,true);

//--- coordonnées du premier point du rattachement de l'objet du Fibonacci
   price1=high[70];
   time1=time[70];
//--- coordonnées du premier point du rattachement de l'objet du Fibonacci
   price2=low[50];
   time2=time[50];

//--- il est temps de créer l'objet Fibonacci
   bool created=ObjectCreate(0,"Fibo",OBJ_FIBO,0,time1,price1,time2,price2);
   if(created) // si l'objet est créé avec succès
     {
      //---établissons la couleur des niveaux Fibo
      ObjectSetInteger(0,"Fibo",OBJPROP_LEVELCOLOR,Blue);
      //--- à propos, combien de niveaux Fibo avons-nous ?
      int levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
      Print("Fibo levels before = ",levels);
      //---déduisons au Journal=>  le numéro du niveau:la valeur la desription_du nive
```

© 2000-2015, MetaQuotes Software Corp.

```
      for(int i=0;i<levels;i++)
        {
         Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
               "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
        }
      //--- essaierons augmenter le nombre de niveaux par une unité
      bool modified=ObjectSetInteger(0,"Fibo",OBJPROP_LEVELS,levels+1);
      if(!modified) // on n'a pas réussi à changer le nombre de niveaux
        {
         Print("On n'a pas réussi à changer le nombre de niveaux dans Fibo, l'erreur "
        }
      //--- juste informons
      Print("Fibo levels after = ",ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS));
      //--- spécifions la valeur pour un nouveau niveau créé
      bool added=ObjectSetDouble(0,"Fibo",OBJPROP_LEVELVALUE,levels,133);
      if(added) // on a réussi à donner la valeur pour le niveau
        {
         Print("On a réussi à établir encore un niveau Fibo");
         //--- ne pas oublier de spécifier la description du niveau
         ObjectSetString(0,"Fibo",OBJPROP_LEVELTEXT,levels,"my level");
         ChartRedraw(0);
         //--- recevrons une valeur actuelle du nombre de niveaux dans l'objet Fibo
         levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
         Print("Fibo levels after adding = ",levels);
         //--- déduisons encore une fois tous les niveaux - simplement pour se persuad
         for(int i=0;i<levels;i++)
           {
            Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
                  "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
           }

        }
      else // échec à la tentative d'augmenter le nombre de niveaux dans l'objet Fibo
        {
         Print("On n'a pas réussi à établir encore un niveau Fibo. L'erreur ",GetLastE
        }
     }
  }
```

Voir aussi

Les types des objets, Les propriétés des objets

© 2000-2015, MetaQuotes Software Corp.

ObjectSetInteger

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type datetime, int, color, bool ou char. Il y a 2 variantes de la fonction.

Etablissement de la valeur de la propriété n'ayant pas le modificateur

```
bool  ObjectSetInteger(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom
   int      prop_id,           // propriété
   long     prop_value         // valeur
   );
```

Etablissement de la valeur de la propriété avec l'indication du modificateur

```
bool  ObjectSetInteger(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom
   int      prop_id,           // propriété
   int      prop_modifier,     // modificateur
   long     prop_value         // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Le modificateur de la propriété indiquée. La plupart des propriétés ne demandent pas le modificateur. Signifie le numéro du niveau dans les instruments de Fibonacci et dans l'objet graphique Andrews' Pitchfork. Le numérotage des niveux commence par le zéro.

prop_value

[in] La valeur de la propriété.

La valeur rendue

Rend true seulement dans le cas si la commande sur le changement des propriétés de l'objet graphique est envoyé avec succès au graphique, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

L'exemple de la création du tableau des couleurs Web

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                                Table of Web Colors|
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             http://www.metaquotes.net |
//+------------------------------------------------------------------+
#define X_SIZE 140      // la largeur de l'objet OBJ_EDIT
#define Y_SIZE 33       // la largeur de l'objet OBJ_EDIT
//+------------------------------------------------------------------+
//| Le tableau des couleurs Web                                      |
//+------------------------------------------------------------------+
color ExtClr[140]=
  {
   clrAliceBlue,clrAntiqueWhite,clrAqua,clrAquamarine,clrAzure,clrBeige,clrBisque,clrB
   clrBlue,clrBlueViolet,clrBrown,clrBurlyWood,clrCadetBlue,clrChartreuse,clrChocolate
   clrCornsilk,clrCrimson,clrCyan,clrDarkBlue,clrDarkCyan,clrDarkGoldenrod,clrDarkGray
   clrDarkMagenta,clrDarkOliveGreen,clrDarkOrange,clrDarkOrchid,clrDarkRed,clrDarkSalm
   clrDarkSlateBlue,clrDarkSlateGray,clrDarkTurquoise,clrDarkViolet,clrDeepPink,clrDee
   clrDodgerBlue,clrFireBrick,clrFloralWhite,clrForestGreen,clrFuchsia,clrGainsboro,cl
   clrGoldenrod,clrGray,clrGreen,clrGreenYellow,clrHoneydew,clrHotPink,clrIndianRed,cl
   clrLavender,clrLavenderBlush,clrLawnGreen,clrLemonChiffon,clrLightBlue,clrLightCora
   clrLightGoldenrod,clrLightGreen,clrLightGray,clrLightPink,clrLightSalmon,clrLightSe
   clrLightSlateGray,clrLightSteelBlue,clrLightYellow,clrLime,clrLimeGreen,clrLinen,cl
   clrMediumAquamarine,clrMediumBlue,clrMediumOrchid,clrMediumPurple,clrMediumSeaGreen
   clrMediumSpringGreen,clrMediumTurquoise,clrMediumVioletRed,clrMidnightBlue,clrMintC
   clrNavajoWhite,clrNavy,clrOldLace,clrOlive,clrOliveDrab,clrOrange,clrOrangeRed,clrC
   clrPaleGreen,clrPaleTurquoise,clrPaleVioletRed,clrPapayaWhip,clrPeachPuff,clrPeru,c
   clrPurple,clrRed,clrRosyBrown,clrRoyalBlue,clrSaddleBrown,clrSalmon,clrSandyBrown,c
   clrSienna,clrSilver,clrSkyBlue,clrSlateBlue,clrSlateGray,clrSnow,clrSpringGreen,clr
   clrThistle,clrTomato,clrTurquoise,clrViolet,clrWheat,clrWhite,clrWhiteSmoke,clrYell
  };
//+------------------------------------------------------------------+
//| La création et l'initialisation de l'objet OBJ_EDIT              |
//+------------------------------------------------------------------+
void CreateColorBox(int x,int y,color c)
  {
//--- générerons le nom pour un nouvel objet selon le nom de la couleur
   string name="ColorBox_"+(string)x+"_"+(string)y;
//--- créons un nouvel objet OBJ_EDIT
   if(!ObjectCreate(0,name,OBJ_EDIT,0,0,0))
     {
      Print("On n'a pas réussi à créer l'objet: '",name,"'");
      return;
     }
//--- spécifions les coordonnées du point du rattachement, la largeur et la hauteur en
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x*X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y*Y_SIZE);
   ObjectSetInteger(0,name,OBJPROP_XSIZE,X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YSIZE,Y_SIZE);
//---définissons la couleur du texte pour l'objet
   if(clrBlack==c) ObjectSetInteger(0,name,OBJPROP_COLOR,clrWhite);
   else            ObjectSetInteger(0,name,OBJPROP_COLOR,clrBlack);
//--- définissons la couleur du fond
   ObjectSetInteger(0,name,OBJPROP_BGCOLOR,c);
//--- définissons le texte de l'objet OBJ_EDIT correspondant à la couleur du fond
   ObjectSetString(0,name,OBJPROP_TEXT,(string)c);
  }
//+------------------------------------------------------------------+
//| La fonction du lancement du script à exécuter                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- créons le tableau des blocs colorés 7x20
   for(uint i=0;i<140;i++)
      CreateColorBox(i%7,i/7,ExtClr[i]);
  }
```

Voir aussi

Les types des objets, Les propriétés des objets

© 2000-2015, MetaQuotes Software Corp.

ObjectSetString

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type string. Il y a 2 variantes de la fonction.

Etablissement de la valeur de la propriété n'ayant pas le modificateur

```
bool  ObjectSetString(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom
   int      prop_id,           // propriété
   string   prop_value         // valeur
   );
```

Etablissement de la valeur de la propriété avec l'indication du modificateur

```
bool  ObjectSetString(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom
   int      prop_id,           // propriété
   int      prop_modifier,     // modificateur
   string   prop_value         // valeur
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Le modificateur de la propriété indiquée. La plupart des propriétés ne demandent pas le modificateur. Signifie le numéro du niveau dans les instruments de Fibonacci et dans l'objet graphique Andrews' Pitchfork. Le numérotage des niveux commence par le zéro.

prop_value

[in] La valeur de la propriété.

La valeur rendue

Rend true seulement dans le cas si la commande sur le changement des propriétés de l'objet graphique est envoyé avec succès au graphique, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

Note

Au changement de nom de l'objet graphique on forme simultanément deux événements, lesquels on

© 2000-2015, MetaQuotes Software Corp.

peut traiter dans l'expert ou l'indicateur par la fonction OnChartEvent():

- l'événement de l'effacement de l'objet avec un vieux nom;
- l'événement de la création de l'objet graphique avec un nouveau nom.

© 2000-2015, MetaQuotes Software Corp.

ObjectGetDouble

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type double. Il y a 2 variantes de la fonction.

1.Rend immédiatement la valeur de propriété.

```
double  ObjectGetDouble(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom de l'objet
   int      prop_id,           // identificateur de la propriété
   int      prop_modifier=0    // modificateur de la propriété, s'il faut
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  ObjectGetDouble(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom de l'objet
   int      prop_id,           // identificateur de la propriété
   int      prop_modifier,     // modificateur de la propriété
   double&  double_var         // acceptons ici la valeur de la propriété
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Le modificateur de la propriété indiquée. Pour la première variante par défaut la valeur du modificateur est égal au 0. La plupart des propriétés ne demandent pas le modificateur. Signifie le numéro du niveau dans les instruments de Fibonacci et dans l'objet graphique Andrews' Pitchfork. Le numérotage des niveux commence par le zéro.

double_var

[out] La variable du type double, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type double pour la première variante de l'appel.

Pour la deuxième variante de l'appel rend true, si cette propriété est soutenue et la valeur était placé à la variable double_var, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

ObjectGetInteger

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type datetime, int, color, bool ou char. Il y a 2 variantes de la fonction.

1. Rend immédiatement la valeur de propriété.

```
long  ObjectGetInteger(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom de l'objet
   int      prop_id,           // identificateur de la propriété
   int      prop_modifier=0    // modificateur de la propriété, s'il faut
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  ObjectGetInteger(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom de l'objet
   int      prop_id,           // identificateur de la propriété
   int      prop_modifier,     // modificateur de la propriété
   long&    long_var           // acceptons ici la valeur de la propriété
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Le modificateur de la propriété indiquée. Pour la première variante par défaut la valeur du modificateur est égal au 0. La plupart des propriétés ne demandent pas le modificateur. Signifie le numéro du niveau dans les instruments de Fibonacci et dans l'objet graphique Andrews' Pitchfork. Le numérotage des niveux commence par le zéro.

long_var

[out] La valeur du type long, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type long pour la première variante de l'appel.

Pour la deuxième variante de l'appel rend true, si cette propriété est soutenue et la valeur était placé à la variable double_var, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

ObjectGetString

Spécifie la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être du type string. Il y a 2 variantes de la fonction..

1. Rend immédiatement la valeur de propriété.

```
string  ObjectGetString(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom de l'objet
   int      prop_id,           // identificateur de la propriété
   int      prop_modifier=0    // modificateur de la propriété, s'il faut
   );
```

2. Rend true ou false en fonction du succès de l'exécution de la fonction. En cas du succès la valeur de la propriété se place à la variable de réception transmise selon la référence par le dernier paramètre.

```
bool  ObjectGetString(
   long     chart_id,          // identificateur du graphique
   string   name,              // nom de l'objet
   int      prop_id,           // identificateur de la propriété
   int      prop_modifier,     // modificateur de la propriété
   string&  string_var         // acceptons ici la valeur de la propriété
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

name

[in] Le nom de l'objet.

prop_id

[in] L'identificateur de la propriété de l'objet. La valeur peut être une des valeurs de l'énumération ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Le modificateur de la propriété indiquée. Pour la première variante par défaut la valeur du modificateur est égal au 0. La plupart des propriétés ne demandent pas le modificateur. Signifie le numéro du niveau dans les instruments de Fibonacci et dans l'objet graphique Andrews' Pitchfork. Le numérotage des niveux commence par le zéro.

string_var

[out] La valeur du type string, acceptant la valeur de la propriété demandée.

La valeur rendue

La valeur du type string pour la première variante de l'appel.

Pour la deuxième variante de l'appel rend true, si cette propriété est soutenue et la valeur était placé à la variable double_var, autrement rend false. Pour recevoir l'information supplémentaire sur l'erreur, il est nécessaire d'appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

Note

Au changement de nom de l'objet graphique on forme simultanément deux événements, lesquels on peut traiter dans l'expert ou l'indicateur par la fonction OnChartEvent():

- l'événement de l'effacement de l'objet avec un vieux nom;
- l'événement de la création de l'objet graphique avec un nouveau nom.

© 2000-2015, MetaQuotes Software Corp.

TextSetFont

Établit la fonte pour la sortie du texte par les méthodes du dessin et rend le résultat de la réussite de cette opération. On utilise par défaut la fonte Arial et la taille -120 (12 pt).

```
bool  TextSetFont(
   const string  name,            // le nom de la fonte ou la voie vers le fichier de
   int           size,            // la taille de la fonte
   uint          flags,           // la combinaison des drapeaux
   int           orientation=0    // l'angle de l'inclinaison du texte
   );
```

Réglages

name

[in] Le nom de la fonte dans le système, ou le nom de la ressource contenant la fonte, ou la voie vers le fichier de la fonte sur le disque.

size

[in] La taille de la fonte, qui peut être spécifiée par les valeurs positives et négatives. Aux valeurs positives la taille du texte déduit ne dépend pas des réglages des tailles des fontes dans le système d'exploitation. Aux valeurs négatives la valeur est spécifiée dans les dixièmes parts du point et la taille du texte dépendra des réglages du système ("l'échelle standard" ou "une grande échelle"). Plus en détail sur la différence entre les modes regardez dans la Note

flags

[in] La combinaison des drapeaux, décrivant le style de la fonte.

orientation

[in] L'angle de l'inclinaison du texte à l'horizontale vers l'axe X, l'unité de mesure est égale 0.1 degrés. C'est-à-dire `orientation=450`signifie l'inclinaison à 45 degrés.

La valeur rendue

Rend true en cas de l'installation réussie de la fonte courante, autrement rend false. Les codes possibles des erreurs:

- ERR_INVALID_PARAMETER(4003) - *name* présente NULL ou "" (la ligne vide),
- ERR_INTERNAL_ERROR(4001) - l'erreur du système d'exploitation (par exemple, la tentative de la création de la fonte inexistante).

Note

Si dans le nom de la fonte est utilisé "::", la fonte est chargée de la ressource EX5. Si le nom de la fonte*name*est indiqué avec l'extansion, la fonte est chargée du fichier, en cela - si la voie commence par "\" ou "/", le fichier est cherché par rapport au catalogue MQL5, autrement il est cherché par rapport à la voie du fichier EX5 qui a appelé la fonction TextSetFont().

La taille de la fonte est spécifiée par les valeurs positives ou négatives, le signe définit la dépendance de la taille du texte des réglages du système d'exploitation (l'échelle de la fonte).

- Si le taille est spécifiée par le nombre positif, à l'affichage de la fonte logique au physique se passe la transformation de la taille en unités physiques de la mesure du dispositif (les pixels) et cette taille correspond à la hauteur des cellules des symboles des fontes accessibles. Il n'est pas recommandé dans les cas où on suppose l'utilisation partagée sur le graphique des textes déduits

© 2000-2015, MetaQuotes Software Corp.

par la fonctionTextOut(), et des textes affichés à l'aide de l'objet graphiqueOBJ_LABEL ("La marque de texte").

- Si la taille est spécifiée par le nombre négatif, la taille indiquée est supposé spécifiée aux dixièmes parts du point logique (la valeur -350 est égal au 35 points logiques) et se divise sur 10, et puis la valeur reçue sera transformée en unités physiques de la mesure du dispositif (les pixels) et correspond à la valeur absolue de la hauteur du symbole des fontes accessibles. Pour recevoir à l'écran le texte de la même taille, comme dans l'objetOBJ_LABEL , prenez la taille de la fonte indiquée aux propriétés de l'objet et multipliez sur -10.

Les drapeaux peuvent être utilisés en forme de la combinaison des drapeaux du style avec un des drapeaux, qui spécifie l'épaisseur de la fonte. Les noms des drapeaux sont plus bas.

Les drapeaux pour spécifier le style du tracé de la fonte

Le drapeau	La description
FONT_ITALIC	L'italique
FONT_UNDERLINE	Le soulignement
FONT_STRIKEOUT	Le barrement

Les drapeaux pour spécifier l'épaisseur de la fonte

Le drapeau
FW_DONTCARE
FW_THIN
FW_EXTRALIGHT
FW_ULTRALIGHT
FW_LIGHT
FW_NORMAL
FW_REGULAR
FW_MEDIUM
FW_SEMIBOLD
FW_DEMIBOLD
FW_BOLD
FW_EXTRABOLD
FW_ULTRABOLD
FW_HEAVY
FW_BLACK

Voir aussi

Les ressources , ResourceCreate(), ResourceSave(), TextOut()

© 2000-2015, MetaQuotes Software Corp.

TextOut

Déduit le texte au tableau d'utilisateur (le tampon) et rend le résultat de la réussite de cette opération. Ce tableau est conçu pour créer le ressourcegraphique.

```
bool  TextOut(
   const string       text,          // le texte déduit
   int                x,             // la coordonnée X
   int                y,             // la coordonnée Y
   uint               anchor,        // le moyen du rattachement
   uint               &data[],       // le tampon pour la sortie
   uint               width,         // la largeur du tampon en points
   uint               height,        // la hauteur du tampon en points
   uint               color,         // la couleur du texte
   ENUM_COLOR_FORMAT  color_format   // le format de la couleur pour la sortie
   );
```

Réglages

text

[in] Le texte déduit, qui sera enregistré au tampon. Seulement une sortie de texte d'une ligne est effectuée.

x

[in] La coordonnée X du point du rattachement du texte.

y

[in] La coordonnée Y du point du rattachement du texte.

anchor

[in] La valeur de l'ensemble de 9 moyens prédéterminés de la disposition du point du rattachement du texte déduit. Est spécifiée par la combinaison de deux drapeaux - le drapeau de l'alignement du texte à l'horizontale et le drapeau de l'alignement du texte selon la verticale. Les noms des drapeaux sont amenés dans la Note.

data[]

[in] Le tampon, où on déduit le texte. Ce tableau est utilisé pour créer le ressource graphique.

width

[in] La largeur du tampon en points (en pixels).

height

[in] La hauteur du tampon en points (en pixels).

color

[in] La couleur du texte.

color_format

[in] Le format de la couleur est spécifié par la valeur de l'énumérationENUM_COLOR_FORMAT.

La valeur rendue

Rend true en cas de l'exécution réussie, autrement rend false.

© 2000-2015, MetaQuotes Software Corp.

Note

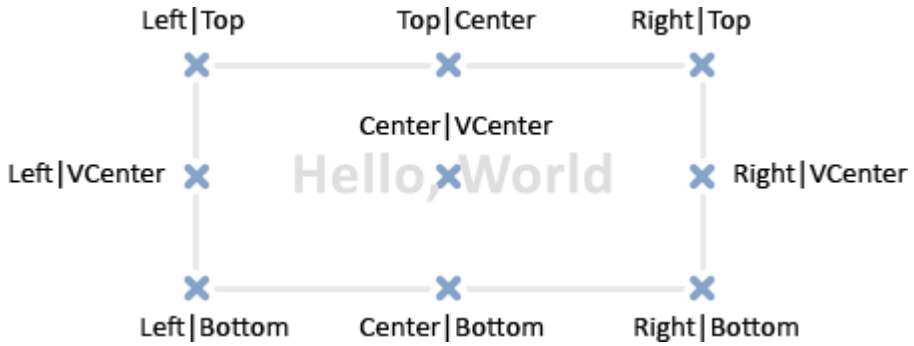
Le moyen du rattachement spécifié par le paramètre `anchor`, qui est la combinaison de deux drapeaux de l'alignement du texte selon la verticale et l'horizontale. Les drapeaux de l'alignement du texte à l'horizontale:

- TA_LEFT - le point du rattachement sur le côté gauche du rectangle limitant
- TA_CENTER - le point du rattachement à l'horizontale se trouve au milieu du rectangle limitant
- TA_RIGHT - le point du rattachement sur le côté droite du rectangle limitant

Les drapeaux de l'alignement du texte selon la verticale:

- TA_TOP - le point du rattachement sur le côté supérieur du rectangle limitant
- TA_VCENTER - le point du rattachement à la verticale se trouve au milieu du rectangle limitant
- TA_BOTTOM - le point du rattachement sur le côté inférieur du rectangle limitant

Les combinaisons possibles des drapeaux et les moyens du rattachement spécifiés par eux sont montrées sur le dessin.



Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//--- la largeur et la hauteur du canvas (la toile, où on fait le dessin)
#define IMG_WIDTH  200
#define IMG_HEIGHT 200
//--- avant le lancement du script montrons la fenêtre avec les paramètres
#property script_show_inputs
//---donnons la possibilité de spécifier le format de la couleur
input ENUM_COLOR_FORMAT clr_format=COLOR_FORMAT_XRGB_NOALPHA;
//--- le tableau (le tampon) pour la rendu
uint ExtImg[IMG_WIDTH*IMG_HEIGHT];
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- créeron l'objet OBJ_BITMAP_LABEL pour le dessin
   ObjectCreate(0,"CLOCK",OBJ_BITMAP_LABEL,0,0,0);
//--- indiquerons le nom de la ressource graphique pour la rendu dans l'objet CLOCK
   ObjectSetString(0,"CLOCK",OBJPROP_BMPFILE,"::IMG");

//--- les variables auxiliaires
   double a;            // l'angle de la flèche
   uint   nm=2700;      // le compteur des minutes
   uint   nh=2700*12;   // le compteur des heures
   uint   w,h;          // les variables pour la réception des tailles des lignes de t
   int    x,y;          // les variables pour le calcul des coordonnées courants du po

//--- tournons les aiguilles des heures dans le cycle infini, jusqu'à ce que le script
   while(!IsStopped())
     {
      //--- le nettoyage du tableau du tampon du dessin des heures
      ArrayFill(ExtImg,0,IMG_WIDTH*IMG_HEIGHT,0);
      //--- l'installation de la fonte pour la rendu des chiffres sur le cadran
      TextSetFont("Arial",-200,FW_EXTRABOLD,0);
      //--- dessinons le cadran
      for(int i=1;i<=12;i++)
        {
         //--- recevrons les tailles de l'heure courante sur le cadran
         TextGetSize(string(i),w,h);
         //--- calculons les coordonnées de l'heure courante sur le cadran
         a=-((i*300)%3600*M_PI)/1800.0;
         x=IMG_WIDTH/2-int(sin(a)*80+0.5+w/2);
         y=IMG_HEIGHT/2-int(cos(a)*80+0.5+h/2);
         //--- la sortie de cette heure sur le cadran au tampon ExtImg[]
         TextOut(string(i),x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,c
        }
      //---maintenant établissons la fonte pour la rendu de la grande aiguille
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nm%3600));
      //--- recevrons les tailles de la grande aiguille
      TextGetSize("----->",w,h);
      //--- calculons les coordonnées de la grande aiguille sur le cadran
      a=-(nm%3600*M_PI)/1800.0;
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- la sortie de la grande aiguille sur le cadran au tampon ExtImg[]
      TextOut("----->",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_f

      //---maintenant établissons la fonte pour la rendu de la petite aiguille
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nh/12%3600));
      TextGetSize("==>",w,h);
      //--- calculons les coordonnées de la petite aiguille sur le cadran
      a=-(nh/12%3600*M_PI)/1800.0;
```

© 2000-2015, MetaQuotes Software Corp.

```
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- la sortie de la petite aiguille sur le cadran au tampon ExtImg[]
      TextOut("==>",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_form

      //--- la mise à jour de la ressource graphique
      ResourceCreate("::IMG",ExtImg,IMG_WIDTH,IMG_HEIGHT,0,0,IMG_WIDTH,clr_format);
      //---la mise à jour obligatoire du graphique
      ChartRedraw();

      //---augmentons les compteurs de l'heure et des minutes
      nm+=60;
      nh+=60;
      //--- tenons une courte pause entre les cadres
      Sleep(10);
     }
//--- supprimons l'objet CLOCK à l'achèvement du travail du script
   ObjectDelete(0,"CLOCK");
//---
  }
```

Voir aussi

Les ressources, ResourceCreate(), ResourceSave(), TextGetSize(), TextSetFont()

© 2000-2015, MetaQuotes Software Corp.

TextGetSize

Rend la largeur et la hauteur de la ligneaux réglages courants de la fonte.

```
bool  TextGetSize(
   const string       text,          // la ligne du texte
   uint&               width,        // la largeur du tampon en points
   uint&               height        // la hauteur du tampon en points
   );
```

Réglages

text

[in] La ligne pour laquelle on reçoit la longueur et la largeur.

width

[out] Le paramètre d'entrée pour la réception de la largeur.

height

[out] Le paramètre d'entrée pour la réception de la hauteur.

La valeur rendue

Rend true en cas de l'exécution réussie, autrement rend false. Les codes possibles des erreurs:

- ERR_INTERNAL_ERROR(4001) - en cas de l'erreur du système d'exploitation.

Voir aussi

Les ressources, ResourceCreate(), ResourceSave(), TextSetFont(), TextOut()

© 2000-2015, MetaQuotes Software Corp.

Les fonctions pour le travail avec les indicateurs techniques

Toutes les fonctions du type iMA, iAC, iMACD, iIchimoku etc., créent une copie de l'indicateur technique correspondant dans la cache globale du terminal de client. Si la copie de l'indicateur avec ces paramètres existe déjà, une nouvelle copie n'est pas créée, mais augmente le compteur des liens à la copie donnée.

Ces fonctions rendent le handle de la copie correspondante de l'indicateur. En utilisant ce handle on peut recevoir par la suite les données calculées par l'indicateur correspondant. On peut copier les données du tampon correspondant (les indicateurs techniques contiennent les données calculées dans les tampons intérieurs, que, en fonction de l'indicateur, peut être de 1 jusqu'à 5) au programme mql5 à l'aide de la fonction CopyBuffer().

On ne peut pas s'adresser aux données de l'indicateur à la fois après sa création, puisque on demande un certain temps sur le calcul des valeurs de l'indicateur, et c'est pour cela c'est mieux de créer les handles des indicateurs à OnInit(). La fonction iCustom() crée l'indicateur correspondant d'utilisateur et à la création réussie rend son handle. Les indicateurs d'utilisateur peuvent contenir jusqu'à 512 des tampons d'indicateur, dont le contenu peut être reçu à l'aide de la fonction CopyBuffer(), en utilisant le handle reçu.

Il y a une méthode universelle de la création de n'importe quel indicateur technique avec l'aide de la fonction IndicatorCreate(). Cette fonction reçoit à titre des paramètres d'entrée:

- le nom du symbole;
- le temps trame;
- le type de l'indicateur créé;
- le nombre de paramètres d'entrée de l'indicateur;
- le tableau du type MqlParam, contenant tous les paramètres nécessaires d'entrée.

La mémoire informatique peut être libérée d'un indicateur qui n'est plus utilisé, en utilisant la fonction IndicatorRelease(), à laquelle le handle d'indicateur est transmise.

Note. L'appel multiple à la fonction de l'indicateur avec les mêmes paramètres dans la limite d'un programme mql5 n'amène pas à l'augmentation multiple du compteur des liens, le compteur sera augmenté seulement une fois sur 1. Il est recommandé de recevoir cependant les handles des indicateurs en fonction OnInit() ou dans le constructeur de la classe, avec l'utilisation ultérieure des handles reçus dans les autres fonctions. Les compteurs des liens diminue à la déinitialisation du programme mql5.

Toutes les fonctions d'indicateur ont au minimum 2 paramètres - le symbole et la période. La valeur du symbole NULL signifie l'instrument courant, la valeur de la période 0 signifie le temps trame courant.

Fonction	Rend le handle de l'indicateur:
iAC	Accelerator Oscillator
iAD	Accumulation/Distribution
iADX	Average Directional Index
iADXWilder	Average Directional Index by Welles Wilder
iAlligator	Alligator

© 2000-2015, MetaQuotes Software Corp.

iAMA	Adaptive Moving Average
iAO	Awesome Oscillator
iATR	Average True Range
iBearsPower	Bears Power
iBands	Bollinger Bands®
iBullsPower	Bulls Power
iCCI	Commodity Channel Index
iChaikin	Chaikin Oscillator
iCustom	l'indicateur d'utilisateur
iDEMA	Double Exponential Moving Average
iDeMarker	DeMarker
iEnvelopes	Envelopes
iForce	Force Index
iFractals	Fractals
iFrAMA	Fractal Adaptive Moving Average
iGatore	Gator Oscillator
iIchimoku	Ichimoku Kinko Hyo
iBWMFI	Market Facilitation Index by Bill Williams
iMomentum	Momentum
iMFI	Money Flow Index
iMA	Moving Average
iOsMA	Moving Average of Oscillator (MACD histogram)
iMACD	Moving Averages Convergence-Divergence
iOBV	On Balance Volume
iSAR	Parabolic Stop And Reverse System
iRSI	Relative Strength Index
iRVI	Relative Vigor Index
iStdDev	Standard Deviation
iStochastic	Stochastic Oscillator
iTEMA	Triple Exponential Moving Average
iTriX	Triple Exponential Moving Averages Oscillator
iWPR	Williams' Percent Range

© 2000-2015, MetaQuotes Software Corp.

iVIDyA	Variable Index DYnamic Average
iVolumes	Volumes

© 2000-2015, MetaQuotes Software Corp.

iAC

Crée au cash global du terminal de client l'indicateur Accelerator Oscillator et rend son handle. Seulement un tampon.

```
int  iAC(
   string           symbol,     // nom du symbole
   ENUM_TIMEFRAMES  period      // période
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeur de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAC.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iAC."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- La construction iAC
#property indicator_label1  "iAC"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAC,               // utiliser iAC
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iAC;          // le type de la fonction
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- les tampons d'indicateur
double iACBuffer[];
double iACColors[];
//--- la variable pour le stockage du handle de l'indicateur iAC
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockons le nombre de valeurs dans l'indicateur Accelerator Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,iACBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iACColors,INDICATOR_COLOR_INDEX);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iAC)
      handle=iAC(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AC);
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       //--- informons sur l'échec et déduisons le numéro d'erreur
       PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iAC pour une pa
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- le travail de l'indicateur est terminé avant terme
       return(INIT_FAILED);
      }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Accelerator Oscil
   short_name=StringFormat("iAC(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iAC
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est le premier lancement des calculs de notre indicateur ou a changé le nom
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iACBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux iACBuffer et iACColors par les valeurs de l'indicateur
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffer(iACBuffer,iACColors,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Accelerator Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iAC         |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // le tampon d'indicateur des valeu
                          double &color_indexes[], // le tampon coloré (pour la conser
                          int ind_handle,          // le handle de l'indicateur iAC
                          int amount               // le nombre de valeurs copiées
                          )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iACBuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAC, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- maintenant nous copions les index des couleurs
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les valeurs des couleurs de l'indicateur
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iAD

Rend le handle de l'indicateur Accumulation/Distribution. Seulement un tampon.

```
int  iAD(
   string               symbol,             // nom du symbole
   ENUM_TIMEFRAMES      period,             // période
   ENUM_APPLIED_VOLUME  applied_volume      // type du volume pour le calcul
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

applied_volume

[in] Le volume utilisé. Peut être l'un des ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAD.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iAD."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iAD
#property indicator_label1  "iAD"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAD,               // utiliser iAD
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iAD;          // le type de la fonction
input ENUM_APPLIED_VOLUME  volumes;                // le volume utilisé
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- le tampon d'indicateur
double         iADBuffer[];
//--- la variable pour stocker le handle de l'indicateur iAD
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//---stockerons le nombre de valeurs dans l'indicateur Accumulation/Distribution
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iADBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iAD)
      handle=iAD(name,period,volumes);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=volumes;
      handle=IndicatorCreate(name,period,IND_AD,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iAD pour la pai
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Accu
   short_name=StringFormat("iAD(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iAD
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- si le tableau iADBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iADBuffer par les valeurs de l'indicateur Accumulation/Di
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArrayFromBuffer(iADBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Accumulation/Distribution
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iAD         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // le tampon d'indicateur de la ligne Acc
                         int ind_handle,     // le handle de l'indicateur iAD
                         int amount          // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iADBuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAD, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iADX

Rend le handle de l'indicateur Average Directional Movement Index.

```
int  iADX(
   string           symbol,          // nom du symbole
   ENUM_TIMEFRAMES  period,          // période
   int              adx_period       // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

adx_period

[in] La période pour le calcul de l'index.

Note

Les numéros des tampons: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iADX.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iADX."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- plot ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADX,              // utiliser iADX
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iADX;         // le type de la fonction
input int                  adx_period=14;          // la période du calcul
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- les tampons d'indicateur
double         ADXBuffer[];
double         DI_plusBuffer[];
double         DI_minusBuffer[];
//--- la variable pour le stockage du handle de l'indicateur iADX
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Average Directional Movement I
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iADX)
      handle=iADX(name,period,adx_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADX,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iADX pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Ave
   short_name=StringFormat("iADX(%s/%s period=%d)",name,EnumToString(period),adx_perio
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iADX
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau ADXBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Average Directional Mover
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur Average Directional Movement I
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iADX        |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool FillArraysFromBuffers(double &adx_values[],      // le tampon d'indicateur de la
                           double &DIplus_values[],   // le tampon d'indicateur pour l
                           double &DIminus_values[],  // le tampon d'indicateur pour l
                           int ind_handle,            // le handle de l'indicateur iAI
                           int amount                 // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau ADXBuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADX, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau DI_plusBuffer par les valeurs du tampon d'indic
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADX, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau DI_plusBuffer par les valeurs du tampon d'indic
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADX, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iADXWilder

Rend le handle de l'indicateur Average Directional Movement Index by Welles Wilder.

```
int  iADXWilder(
   string           symbol,          // nom du symbole
   ENUM_TIMEFRAMES  period,          // période
   int              adx_period       // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

adx_period

[in] La période pour le calcul de l'index.

Note

Les numéros des tampons: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   iADXWilder.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iADXWilder
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- plot ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADXWilder,         // utiliser iADXWilder
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iADXWilder;   // le type de la fonction
input int                  adx_period=14;          // la période du calcul
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- les tampons d'indicateur
double         ADXBuffer[];
double         DI_plusBuffer[];
double         DI_minusBuffer[];
//--- la variable pour stocker le handle de l'indicateur iADXWilder
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Average Directional Movement I
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iADXWilder)
      handle=iADXWilder(name,period,adx_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADXW,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iADXWilder pou
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Ave
   short_name=StringFormat("iADXWilder(%s/%s period=%d)",name,EnumToString(period),ad
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iADXWilder
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau ADXBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Average Directional Mover
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur Average Directional Movement I
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iADXWilder  |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool FillArraysFromBuffers(double &adx_values[],      // le tampon d'indicateur de la
                           double &DIplus_values[],   // le tampon d'indicateur pour l
                           double &DIminus_values[],  // le tampon d'indicateur pour l
                           int ind_handle,            // le handle de l'indicateur iAI
                           int amount                 // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau ADXBuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADXWilder,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau DI_plusBuffer par les valeurs du tampon d'indic
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADXWilder,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau DI_plusBuffer par les valeurs du tampon d'indic
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iADXWilder,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iAlligator

Rend le handle de l'indicateur Alligator.

```
int  iAlligator(
   string               symbol,           // nom du symbole
   ENUM_TIMEFRAMES      period,           // période
   int                  jaw_period,       // période pour le calcul des mâchoires
   int                  jaw_shift,        // décalage horizontal des mâchoires
   int                  teeth_period,     // période pour le calcul des dents
   int                  teeth_shift,      // décalage horizontal des dents
   int                  lips_period,      // période pour le calcul des lèvres
   int                  lips_shift,       // décalage horizontal des lèvres
   ENUM_MA_METHOD       ma_method,        // type de lissage
   ENUM_APPLIED_PRICE   applied_price     // type de prix ou handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

jaw_period

[in] La période de la prise de moyenne de la ligne bleue (la mâchoire d'alligator).

jaw_shift

[in] Le décalage de la ligne bleue par rapport au graphique du prix.

teeth_period

[in] La période de la prise de moyenne de la ligne rouge (les dents d'alligator).

teeth_shift

[in] Le déplacement de la ligne rouge par rapport au graphique du prix.

lips_period

[in] La période de la prise de moyenne de la ligne verte (les lèvres d'alligator).

lips_shift

[in] Le déplacement de la ligne verte par rapport au graphique du prix.

ma_method

[in] La méthode de la prise de moyenne.Peut être chacune des valeurs de l'énumération ENUM_MA_METHOD.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

© 2000-2015, MetaQuotes Software Corp.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - GATORJAW_LINE, 1 - GATORTEETH_LINE, 2 - GATORLIPS_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Demo_iAlligator.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iAlligator
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans l'Alligator standard."

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- plot Jaws
#property indicator_label1  "Jaws"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot Teeth
#property indicator_label2  "Teeth"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot Lips
#property indicator_label3  "Lips"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrLime
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAlligator,         // utiliser iAlligator
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iAlligator;   // le type de la fonction
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
input int                  jaw_period=13;          // la période pour la ligne des Mâc
input int                  jaw_shift=8;            // le décalage de la ligne des Mâch
input int                  teeth_period=8;         // la période pour la ligne des Der
input int                  teeth_shift=5;          // le décalage de la ligne des Mâch
input int                  lips_period=5;          // la période pour la ligne des Lèv
input int                  lips_shift=3;           // le décalage de la ligne des Lèvr
input ENUM_MA_METHOD       MA_method=MODE_SMMA;    // la méthode de la prise de moyenn
input ENUM_APPLIED_PRICE   applied_price=PRICE_MEDIAN;// le type de prix utilisé pour
//--- les tampons d'indicateur
double         JawsBuffer[];
double         TeethBuffer[];
double         LipsBuffer[];
//--- la variable pour stocker le handle de l'indicateur iAlligator
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Alligator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
//--- spécifions le décalage pour chaque ligne
   PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iAlligator)
      handle=iAlligator(name,period,jaw_period,jaw_shift,teeth_period,
                        teeth_shift,lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[8];
     //--- les périodes et les décalages des lignes de l'Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
//---le type du lissage
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
//--- le type du prix
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
//--- créons le handle
      handle=IndicatorCreate(name,period,IND_ALLIGATOR,8,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iAlligator pou
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Alli
   short_name=StringFormat("iAlligator(%s/%s, %d,%d,%d,%d,%d,%d)",name,EnumToString(pe
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
```

© 2000-2015, MetaQuotes Software Corp.

```
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iAlligator
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau JawsBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur Alligator
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer,l
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
```

© 2000-2015, MetaQuotes Software Corp.

```
                              values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Alligator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iAlligator  |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &jaws_buffer[],  // le tampon d'indicateur pour la l
                           int j_shift,            // le décalage de la ligne des Mâch
                           double &teeth_buffer[], // le tampon d'indicateur pour la l
                           int t_shift,            // le décalage de la ligne des Dent
                           double &lips_buffer[],  // le tampon d'indicateur pour la l
                           int l_shift,            // le décalage de la ligne des Lèvr
                           int ind_handle,         // le handle de l'indicateur iAlliç
                           int amount              // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau JawsBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAlligator,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau TeethBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAlligator,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau LipsBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAlligator,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iAMA

Rend le handle de l'indicateur Adaptive Moving Average. Seulement un tampon.

```
int  iAMA(
   string               symbol,             // nom du symbole
   ENUM_TIMEFRAMES      period,             // période
   int                  ama_period,         // période AMA
   int                  fast_ma_period,     // période de la glissante rapide
   int                  slow_ma_period,     // période de la glissante lente
   int                  ama_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_APPLIED_PRICE   applied_price       // type du prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ama_period

[in] La période du calcul, dont on calcule le coefficient d'efficacité.

fast_ma_period

[in] La période rapide pour le calcul du coefficient de l'effacement aux moments du marché rapide.

slow_ma_period

[in] La période lente pour le calcul du coefficient de l'effacement dans l'absence de la tendance.

ama_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                    Demo_iAMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iAMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans AMA standard."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iAMA
#property indicator_label1  "iAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAMA,              // utiliser iAMA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iAMA;          // le type de la fonction
input string               symbol=" ";              // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;   // le temps trame
input int                  ama_period=15;           // la période pour le calcul
input int                  fast_ma_period=2;        // la période de la glissante rapi
input int                  slow_ma_period=30;       // la période de la glissante lent
input int                  ama_shift=0;             // le décalage par l'horizontale
input ENUM_APPLIED_PRICE   applied_price;           // le type du prix
//--- le tampon d'indicateur
double         iAMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iAMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Adaptive Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iAMABuffer,INDICATOR_DATA);
//--- spécifions le décalage
   PlotIndexSetInteger(0,PLOT_SHIFT,ama_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iAMA)
      handle=iAMA(name,period,ama_period,fast_ma_period,slow_ma_period,ama_shift,appli
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[5];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ama_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=fast_ma_period;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slow_ma_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ama_shift;
      //--- le type du prix
      pars[4].type=TYPE_INT;
      pars[4].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_AMA,5,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iAMA pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Adaptive Moving Av
   short_name=StringFormat("iAMA(%s/%s,%d,%d,%d,d)",name,EnumToString(period),ama_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iAlligator
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iADBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur Alligator
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArrayFromBuffer(iAMABuffer,ama_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
```

© 2000-2015, MetaQuotes Software Corp.

```
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Accelerator Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iAMA          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // le tampon d'indicateur de la ligne
                         int a_shift,           // le décalage de la ligne AMA
                         int ind_handle,        // le handle de l'indicateur iAMA
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iAMABuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,-a_shift,amount,ama_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAlligator,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iAO

Rend le handle de l'indicateur Awesome oscillator. Seulement un tampon.

```
int  iAO(
   string           symbol,     // nom du symbole
   ENUM_TIMEFRAMES  period      // période
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAO.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iAO."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- la construction iAO
#property indicator_label1  "iAO"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen,clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAO,               // utiliser iAO
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iAO;          // le type de la fonction
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- les tampons d'indicateur
double         iAOBuffer[];
double         iAOColors[];
//--- la variable pour stocker le handle de l'indicateur iAO
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Awesome Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,iAOBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iAOColors,INDICATOR_COLOR_INDEX);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iAO)
      handle=iAO(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AO);
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- informons sur l'échec et déduisons le numéro d'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iAO pour la pai
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Awesome Oscillator
   short_name=StringFormat("iAO(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iAO
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nom
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iAOBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux iAOBuffer et iAOColors par les valeurs de l'indicateur
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffer(iAOBuffer,iAOColors,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Accelerator Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iAO         |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // le tampon d'indicateur des valeu
                          double &color_indexes[], // le tampon coloré (pour la conse
                          int ind_handle,          // le handle de l'indicateur iAO
                          int amount               // le nombre de valeurs copiées
                          )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iAOBuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iAO, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- maintenant nous copions les index des couleurs
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les valeurs des couleurs de l'indicateur
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iATR

Rend le handle de l'indicateur Average True Range. Seulement un tampon.

```
int  iATR(
   string           symbol,        // nom du symbole
   ENUM_TIMEFRAMES  period,        // période
   int              ma_period      // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iATR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iATR."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iATR
#property indicator_label1  "iATR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iATR,// utiliser iATR
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input int                  atr_period=14;          // la période pour le calcul
input Creation             type=Call_iATR;          // le type de la fonction
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- le tampon d'indicateur
double         iATRBuffer[];
//--- la variable pour le stockage du handle de l'indicateur iAC
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Average True Range
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iATRBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iATR)
      handle=iATR(name,period,atr_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=atr_period;
      handle=IndicatorCreate(name,period,IND_ATR,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro d'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iATR pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Ave
   short_name=StringFormat("iATR(%s/%s, period=%d)",name,EnumToString(period),atr_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iATR
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- si le tableau iATRBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iATRBuffer  par les valeurs de l'indicateur Average True
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iATRBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Accelerator Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iATR          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs ATR
                         int ind_handle,    // le handle de l'indicateur iATR
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iATRBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iATR, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iBearsPower

Rend le handle de l'indicateur Bears Power. Seulement un tampon.

```
int  iBearsPower(
   string              symbol,           // nom du symbole
   ENUM_TIMEFRAMES     period,           // période
   int                 ma_period,        // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBearsPower.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iBearsPowe
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iBearsPower
#property indicator_label1  "iBearsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBearsPower,       // utiliser iBearsPower
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iBearsPower;  // le type de la fonction
input int                  ma_period=13;           // la période de la glissante
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- le tampon d'indicateur
double         iBearsPowerBuffer[];
//--- a variable pour stocker le handle de l'indicateur iBearsPower
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Bears Power
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iBearsPowerBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iBearsPower)
      handle=iBearsPower(name,period,ma_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
      MqlParam pars[1];
      //--- la période de la glissante
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BEARS,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro d'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iBearsPower pou
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Bears Powera été c
   short_name=StringFormat("iBearsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iBearsPower
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      //--- si le tableau iATRBuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iBearsPowerBuffer par les valeurs de l'indicateur Bears P
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iBearsPowerBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur Bears Power
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iBearsPower   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Bear
                         int ind_handle,    // le handle de l'indicateur iBearsPower
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iBearsPowerBuffer par les valeurs du tampon d'i
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iBearsPower,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iBands

Rend le handle de l'indicateur Bollinger Bands®.

```
int  iBands(
   string               symbol,           // nom du symbole
   ENUM_TIMEFRAMES      period,           // période
   int                  bands_period,     // période pour le calcul de la moyenne ligr
   int                  bands_shift,      // décalage de l'indicateur à l'horizontale
   double               deviation,        // nombre de divergences standards
   ENUM_APPLIED_PRICE   applied_price     // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

bands_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

bands_shift

[in] Le décalage l'indicateur par rapport au graphique des prix.

deviation

[in] La divergence de la ligne principale.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - BASE_LINE, 1 - UPPER_BAND, 2 - LOWER_BAND

Exemple:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iBands.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iBands."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- la construction Upper
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Lower
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrMediumSeaGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- la construction Middle
#property indicator_label3  "Middle"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrMediumSeaGreen
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBands,             //utiliser iBands
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iBands;          // le type de la fonction
input int                  bands_period=20;           // la période de la glissante mo
input int                  bands_shift=0;             // le décalage
input double               deviation=2.0;             // le nombre de divergences star
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- les tampons d'indicateur
double         UpperBuffer[];
double         LowerBuffer[];
```

© 2000-2015, MetaQuotes Software Corp.

```
double         MiddleBuffer[];
//--- la variable pour stocker le handle de l'indicateur iBands
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Bollinger Bands
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,MiddleBuffer,INDICATOR_DATA);
//--- spécifions le décalage pour chaque ligne
   PlotIndexSetInteger(0,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,bands_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iBands)
      handle=iBands(name,period,bands_period,bands_shift,deviation,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période de la glissante
      pars[0].type=TYPE_INT;
      pars[0].integer_value=bands_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=bands_shift;
      //--- le nombre de divergences standards
      pars[2].type=TYPE_DOUBLE;
      pars[2].double_value=deviation;
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_BANDS,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iBands pour la
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Bollinger Bands a
   short_name=StringFormat("iBands(%s/%s, %d,%d,%G,%s)",name,EnumToString(period),
                           bands_period,bands_shift,deviation,EnumToString(applied_pri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iBands
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      //--- si la taille des tableaux d'indicateur est plus grand que des valeurs dans
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Bollinger Bands
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(MiddleBuffer,UpperBuffer,LowerBuffer,bands_shift,handle,v
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Bollinger Bands
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iBands      |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &base_values[],     // le tampon d'indicateur de la
                           double &upper_values[],    // le tampon d'indicateur de la
                           double &lower_values[],    // le tampon d'indicateur de la
                           int shift,                 // le décalage
                           int ind_handle,            // le handle de l'indicateur iBa
                           int amount                 // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau MiddleBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,0,-shift,amount,base_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("n'a pas réussi à copier les données de l'indicateur iBands, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- remplissons la partie du tableau UpperBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("n'a pas réussi à copier les données de l'indicateur iBands, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau LowerBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,2,-shift,amount,lower_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("n'a pas réussi à copier les données de l'indicateur iBands, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iBullsPower

Rend le handle de l'indicateur Bulls Power. Seulement un tampon.

```
int  iBullsPower(
   string              symbol,           // nom du symbole
   ENUM_TIMEFRAMES     period,           // période
   int                 ma_period,        // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBullsPower.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iBullsPowe
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iBullsPower
#property indicator_label1  "iBullsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBullsPower,       // utiliser iBullsPower
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iBullsPower;  // le type de la fonction
input int                  ma_period=13;           // la période de la glissante
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
//--- le tampon d'indicateur
double         iBullsPowerBuffer[];
//--- la variable pour stocker le handle de l'indicateur iBullsPower
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Bulls Power
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iBullsPowerBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iBullsPower)
      handle=iBullsPower(name,period,ma_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
      MqlParam pars[1];
      //--- la période de la glissante
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BULLS,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro d'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iBullsPower pou
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Bulls Power a été
   short_name=StringFormat("iBullsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iBullsPower
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      //--- si le tableau iBullsPowerBuffer est plus grand, que les valeurs dans l'ind
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iBullsPowerBuffer  par les valeurs de l'indicateur Bulls
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iBullsPowerBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Bulls Power
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iBullsPower   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Bull
                         int ind_handle,    // le handle de l'indicateur iBullsPower
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iBullsPowerBuffer par les valeurs du tampon d'i
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iBullsPower,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

iCCI

Rend le handle de l'indicateur Commodity Channel Index. Seulement un tampon.

```
int  iCCI(
   string               symbol,           // nom du symbole
   ENUM_TIMEFRAMES      period,           // période
   int                  ma_period,        // période de la prise de moyenne
   ENUM_APPLIED_PRICE   applied_price     // type du prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iCCI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iCCI."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_plots   1
//--- la construction iCCI
#property indicator_label1  "iCCI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iCCI,              // utiliser iCCI
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iCCI;               // le type de la fonction
input int                  ma_period=14;                 // la période de la glissante
input ENUM_APPLIED_PRICE   applied_price=PRICE_TYPICAL;  // le type du prix
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- le tampon d'indicateur
double         iCCIBuffer[];
//--- la variable pour stocker le handle de l'indicateur iCCI
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Commodity Channel Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iCCIBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iCCI)
      handle=iCCI(name,period,ma_period,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le type du prix
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_CCI,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iCCI pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Bollinger Bands® a
   short_name=StringFormat("iCCI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//--- le nombre de valeurs copiées de l'indicateur iCCI
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iATRBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iCCIBuffer   par les valeurs de l'indicateur Commodity Ch
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iCCIBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Commodity Channel Index
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iCCI          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Comm
                         int ind_handle,    // le handle de l'indicateur iCCI
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
```

© 2000-2015, MetaQuotes Software Corp.

```
   ResetLastError();
//--- remplissons la partie du tableau iCCIBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iCCI, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iChaikin

Rend le handle de l'indicateur Chaikin Oscillator. Seulement un tampon.

```
int  iChaikin(
   string               symbol,             // nom du symbole
   ENUM_TIMEFRAMES      period,             // période
   int                  fast_ma_period,     // période rapide
   int                  slow_ma_period,     // période lente
   ENUM_MA_METHOD       ma_method,          // type de lissage
   ENUM_APPLIED_VOLUME  applied_volume      // volume utilisé
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

fast_ma_period

[in] La période rapide de la prise de moyenne pour le calcul de l'indicateur.

slow_ma_period

[in] La période lente de la prise de moyenne pour le calcul de l'indicateur.

ma_method

[in] Le type de la prise de moyenne. Peut être chacune des constantes de la prise de moyenne ENUM_MA_METHOD.

applied_volume

[in] Le volume utilisé. Peut être chacun de l'énumération ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iChaikin.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
```

© 2000-2015, MetaQuotes Software Corp.

```
#property description "des tampons d'indicateur pour l'indicateur technique iChaikin."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iChaikin
#property indicator_label1  "iChaikin"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iChaikin,          // utiliser iChaikin
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation              type=Call_iChaikin;          // le type de la fonction
input int                   fast_ma_period=3;            // la période rapide
input int                   slow_ma_period=10;           // la période lente
input ENUM_MA_METHOD        ma_method=MODE_EMA;          // le type du lissage
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;  // le type du volume
input string                symbol=" ";                  // le symbole
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // le temps trame
//--- le tampon d'indicateur
double         iChaikinBuffer[];
//--- la variable pour stocker le handle de l'indicateur iChaikin
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Chaikin Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iChaikinBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iChaikin)
      handle=iChaikin(name,period,fast_ma_period,slow_ma_period,ma_method,applied_volu
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période rapide
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ma_period;
      //--- la période lente
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ma_period;
      //--- le type du lissage
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- le type du volume
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_CHAIKIN,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iChaikin pour l
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Chaikin Oscillato
   short_name=StringFormat("iChaikin(%s/%s, %d, %d, %s, %s)",name,EnumToString(period)
                           fast_ma_period,slow_ma_period,
                           EnumToString(ma_method),EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iChaikin
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iChaikinBuffer est plus grand, que les valeurs dans l'indica
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iChaikinBuffer  par les valeurs de l'indicateur Chaikin C
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iChaikinBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Chaikin Oscillator
```

© 2000-2015, MetaQuotes Software Corp.

```
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iChaikin      |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Chai
                         int ind_handle,    // le handle de l'indicateur iChaikin
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iChaikinBuffer par les valeurs du tampon d'indi
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iChaikin, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iCustom

Rend le handle de l'indicateur indiqué d'utilisateur.

```
int  iCustom(
   string           symbol,     // nom du symbole
   ENUM_TIMEFRAMES  period,     // période
   string           name        // dossier/le nom de_l'indicateur d'utilisateur
   ...                          // liste des paramètres d'entrée de l'indicateur
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

name

[in] Le nom de l'indicateur d'utilisateur contenant le chemin relativement le directoire radical des indicateurs (MQL5/Indicators/). Si l'indicateur se trouve au sous-répertoire, par exemple, dans MQL5/Indicators/**Examples**, son nom doit être spécifié comme - "**Examples**\\le nom_de l'indicateur" (l'indication de l'inverse double slash au lieu de slash simple comme le délimiteur).

...

[in] les paramètres input de l'indicateur d'utilisateur divisés par les virgules. Le type et l'héritage des paramètres doit correspondre. Si les paramètres ne sont pas indiqués, on utilise les valeurs par défaut.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE.

Note

L'indicateur d'utilisateur doit être compilé (le fichier avec l'extension EX5) et se trouver dans le directoire *MQL5/Indicators* du terminal de client ou le sous-répertoire inclus.

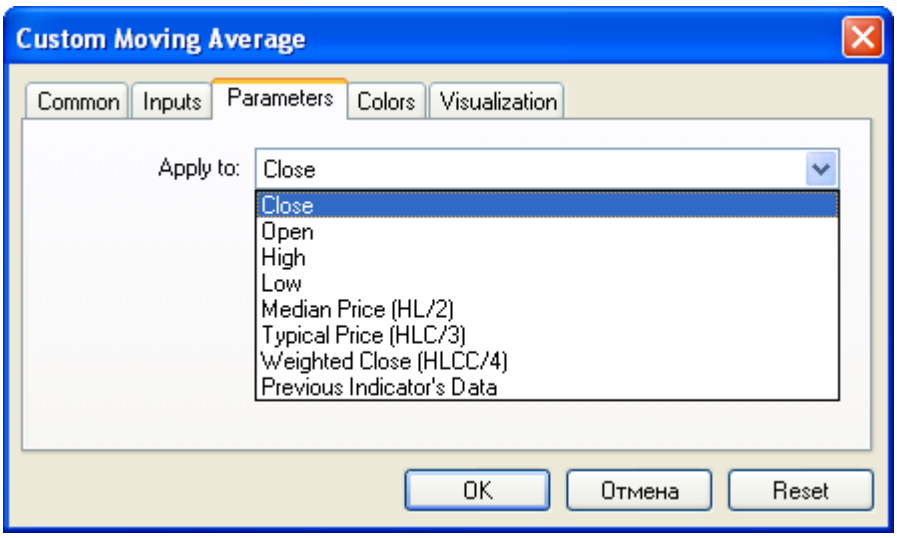
Les indicateurs nécessaires au test sont définis automatiquement de l'appel des fonctions, iCustom(), si le paramètre correspondant est spécifié par la ligne constante. Pour les autres cas (l'utilisation de la fonction IndicatorCreate() ou l'utilisation de la ligne non constante dans le paramètre qui spécifie le nom de l'indicateur) la propriété donnée #property tester_indicator est nécessaire:

```
#property tester_indicator "indicator_name.ex5"
```

Si dans l'indicateur on utilise la première forme de l'appel, au lancement de l'indicateur d'utilisateur sur l'onglet "Parameters" on peut indiquer supplémentairement sur quelles données il sera calculé. Si le paramètre "Apply to" n'est pas choisi évidemment, par défaut le calcul est produit selon les

© 2000-2015, MetaQuotes Software Corp.

valeurs "Close".



A l'appel de l'indicateur d'utilisateur du programme mql5 le paramètre Applied_Price ou le handle d'un autre indicateur doit être transmis le dernier après toutes les variables d'entrée prévues par l'indicateur d'utilisateur.

Voir aussi

Propriétés des programmes, Accès aux séries temporelles et les données des indicateurs, IndicatorCreate(), IndicatorRelease()

Exemple:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int MA_Period=21;
input int MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
//--- indicator buffers
double         Label1Buffer[];
//--- handle de l'indicateur d'utilisateur Custom Moving Average.mq5
int MA_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
```

© 2000-2015, MetaQuotes Software Corp.

```
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
   ResetLastError();
   MA_handle=iCustom(NULL,0,"Examples\\Custom Moving Average",
                     MA_Period,
                     MA_Shift,
                     MA_Method,
                     PRICE_CLOSE // calculons selon les prix de la clôture
                     );
   Print("MA_handle = ",MA_handle,"  error = ",GetLastError());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- copions les valeurs de l'indicateur Custom Moving Average à notre tampon d'indic
   int copy=CopyBuffer(MA_handle,0,0,rates_total,Label1Buffer);
   Print("copy = ",copy,"    rates_total = ",rates_total);
//--- si la tentative est ratée - signalez-le
   if(copy<=0)
      Print("La tentative ratée de recevoir les valeurs de l'indicateur  Custom Moving
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

iDEMA

Rend le handle de l'indicateur Double Exponential Moving Average. Seulement un tampon.

```
int  iDEMA(
   string               symbol,            // nom du symbole
   ENUM_TIMEFRAMES      period,            // période
   int                  ma_period,         // période de la prise de moyenne
   int                  ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_APPLIED_PRICE   applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iDEMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iDEMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
```

© 2000-2015, MetaQuotes Software Corp.

```
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iDEMA
#property indicator_label1  "iDEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDEMA,             // utiliser iDEMA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iDEMA;           // le type de la fonction
input int                  ma_period=14;              // la période de la glissante mo
input int                  ma_shift=0;                // le décalage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iDEMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iDEMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Double Exponential Moving Aver
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iDEMABuffer,INDICATOR_DATA);
   //---spécifions le décalage
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
```

© 2000-2015, MetaQuotes Software Corp.

```
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iDEMA)
      handle=iDEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[3];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du prix
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_DEMA,3,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iDEMA pour la p
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Double Exponential
   short_name=StringFormat("iDEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iDEMA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iDEMABuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iDEMABuffer  par les valeurs de l'indicateur Double Expor
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iDEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//---retiendrons le nombre de valeurs dans l'indicateur Double Exponential Moving Aver
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iDEMA         |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Doub
                         int shift,         // le décalage
                         int ind_handle,    // le handle de l'indicateur iDEMA
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iDEMABuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iDEMA, le co
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera con
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iDeMarker

Rend le handle de l'indicateur DeMarker. Seulement un tampon.

```
int  iDeMarker(
   string           symbol,        // nom du symbole
   ENUM_TIMEFRAMES  period,        // période
   int              ma_period      // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Demo_iDeMarker.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iDeMarker."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul"
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iDeMarker
#property indicator_label1  "iDeMarker"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  0.3
#property indicator_level2  0.7
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDeMarker,         // utiliser iDeMarker
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iDeMarker;       // le type de la fonction
input int                  ma_period=14;              // la période de la glissante
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iDeMarkerBuffer[];
//--- la variable pour stocker le handle de l'indicateur iDeMarker
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur DeMarker
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iDeMarkerBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iDeMarker)
      handle=iDeMarker(name,period,ma_period);
```

© 2000-2015, MetaQuotes Software Corp.

```
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_DEMARKER,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iDeMarker pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur DeMa
   short_name=StringFormat("iDeMarker(%s/%s, period=%d)",name,EnumToString(period),ma_
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iDeMarker
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iDEMABuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iDeMarkerBuffer par les valeurs de l'indicateur DeMarker
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iDeMarkerBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur DeMarker
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iDeMarker     |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs DeMa
                         int ind_handle,    // le handle de l'indicateur iDeMarker
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iDeMarkerBuffer par les valeurs du tampon d'ind
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iDeMarker, l
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
```

© 2000-2015, MetaQuotes Software Corp.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iEnvelopes

Rend le handle de l'indicateur Envelopes.

```
int  iEnvelopes(
   string               symbol,           // nom du symbole
   ENUM_TIMEFRAMES      period,           // période
   int                  ma_period,        // période pour le calcul de la ligne moyenn
   int                  ma_shift,         // décalage de l'indicateur à l'horizontale
   ENUM_MA_METHOD       ma_method,        // type de lissage
   ENUM_APPLIED_PRICE   applied_price,    // type de prix ou le handle
   double               deviation         // divergence des frontières de la ligne moy
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne de la ligne principale de l'indicateur.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique des prix.

ma_method

[in] La méthode de la prise de moyenne. Peut être chacune des valeurs de l'énumération ENUM_MA_METHOD.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

deviation

[in] La divergence de la ligne principale en pourcentage.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - UPPER_LINE, 1 - LOWER_LINE.

Exemple:

```
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//|                                                Demo_iEnvelopes.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iEnvelopes
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- la construction Upper
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Lower
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iEnvelopes,        // utiliser iEnvelopes
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iEnvelopes;      // le type de la fonction
input int                  ma_period=14;              // la période de la glissante
input int                  ma_shift=0;                // le décalage
input ENUM_MA_METHOD       ma_method=MODE_SMA;        // le type du lissage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input double               deviation=0.1;             // le divergence des frontières
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         UpperBuffer[];
double         LowerBuffer[];
//--- la variable pour stocker le handle de l'indicateur iEnvelopes
```

© 2000-2015, MetaQuotes Software Corp.

```
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Envelopes
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
//--- spécifions le décalage pour chaque ligne
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iEnvelopes)
      handle=iEnvelopes(name,period,ma_period,ma_shift,ma_method,applied_price,deviati
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[5];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du lissage
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      //--- le type du prix
```

© 2000-2015, MetaQuotes Software Corp.

```
      pars[4].type=TYPE_DOUBLE;
      pars[4].double_value=deviation;
      handle=IndicatorCreate(name,period,IND_ENVELOPES,5,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iEnvelopes pou
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Enve
   short_name=StringFormat("iEnvelopes(%s/%s, %d, %d, %s,%s, %G)",name,EnumToString(pe
   ma_period,ma_shift,EnumToString(ma_method),EnumToString(applied_price),deviation);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iEnvelopes
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- si le tableau UpperBuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux UpperBuffer et LowerBuffer par les valeurs de l'indicat
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArraysFromBuffers(UpperBuffer,LowerBuffer,ma_shift,handle,values_to_copy))
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Envelopes
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iEnvelopes  |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &upper_values[],    // le tampon d'indicateur de la
                           double &lower_values[],    // le tampon d'indicateur de la
                           int shift,                 // le décalage
                           int ind_handle,            // le handle de l'indicateur iEr
                           int amount                 // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau UpperBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,-shift,amount,upper_values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iEnvelopes,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- remplissons la partie du tableau LowerBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,1,-shift,amount,lower_values)<0)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iEnvelopes,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iForce

Rend le handle de l'indicateur Force Index. Seulement un tampon.

```
int  iForce(
   string               symbol,            // nom du symbole
   ENUM_TIMEFRAMES      period,            // période
   int                  ma_period,         // période de la prise de moyenne
   ENUM_MA_METHOD       ma_method,         // type de lissage
   ENUM_APPLIED_VOLUME  applied_volume     // type du volume pour le calcul
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

ma_method

[in] La méthode de la prise de moyenne. Peut être chacune des valeurs de l'énumération ENUM_MA_METHOD.

applied_volume

[in] Le volume utilisé. Peut être chacun des valeurs de l'énumération ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iForce.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iForce."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
```

© 2000-2015, MetaQuotes Software Corp.

```
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iForce
#property indicator_label1  "iForce"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iForce,            // utiliser iForce
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iForce;             // le type de la fonction
input int                  ma_period=13;                 // la période de la prise de
input ENUM_MA_METHOD       ma_method=MODE_SMA;           // le type du lissage
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // le type du volume
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- le tampon d'indicateur
double         iForceBuffer[];
//--- la variable pour stocker le handle de l'indicateur iForce
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Force
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iForceBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iForce)
      handle=iForce(name,period,ma_period,ma_method,applied_volume);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[3];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le type du lissage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_method;
      //--- le type du volume
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_volume;
      //--- le type du prix
      handle=IndicatorCreate(name,period,IND_FORCE,3,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iForce pour la
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Forc
   short_name=StringFormat("iForce(%s/%s, %d, %s, %s)",name,EnumToString(period),
                           ma_period,EnumToString(ma_method),EnumToString(applied_volu
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iForce
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iForceBuffer est plus grand, que les valeurs dans l'indicate
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iForceBuffer par les valeurs de l'indicateur Force
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iForceBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Force
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iForce        |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Forc
                         int ind_handle,    // le handle de l'indicateur iForce
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iForceBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iForce, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera con
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iFractals

Rend le handle de l'indicateur Fractals.

```
int  iFractals(
   string           symbol,     // nom du symbole
   ENUM_TIMEFRAMES  period      // période
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - UPPER_LINE, 1 - LOWER_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Demo_iFractals.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iFractals."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul"
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- la construction FractalUp
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_label1  "FractalUp"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
//--- la construction FractalDown
#property indicator_label2  "FractalDown"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFractals,         // utiliser iFractals
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iFractals;          // le type de la fonction
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- les tampons d'indicateur
double         FractalUpBuffer[];
double         FractalDownBuffer[];
//--- la variable pour stocker le handle de l'indicateur iFractals
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Fractals
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,FractalUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,FractalDownBuffer,INDICATOR_DATA);
//--- spécifions les codes par le symbole de l'ensemble Wingdings pour les propriétés
   PlotIndexSetInteger(0,PLOT_ARROW,217); // la flèche en haut
   PlotIndexSetInteger(1,PLOT_ARROW,218); // la flèche en bas
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iFractals)
      handle=iFractals(name,period);
   else
      handle=IndicatorCreate(name,period,IND_FRACTALS);
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iFractals pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Frac
   short_name=StringFormat("iFractals(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iFractals
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau FractalUpBuffer est plus grand, que les valeurs dans l'indic
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux FractalUpBuffer et FractalDownBuffer par les valeurs de
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArraysFromBuffers(FractalUpBuffer,FractalDownBuffer,handle,values_to_copy))
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Fractals
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iFractals   |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &up_arrows[],        // le tampon d'indicateur des f
                           double &down_arrows[],      // le tampon d'indicateur des f
                           int ind_handle,             // le handle de l'indicateur iF
                           int amount                  // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//---remplissons la partie du tableau FractalUpBuffer par les valeurs du tampon d'indi
   if(CopyBuffer(ind_handle,0,0,amount,up_arrows)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iFractals au
                  GetLastError());
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- remplissons la partie du tableau FractalDownBuffer par les valeurs du tampon d'i
   if(CopyBuffer(ind_handle,1,0,amount,down_arrows)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iFractals da
                  GetLastError());
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iFrAMA

Rend le handle de l'indicateur Fractal Adaptive Moving Average. Seulement un tampon.

```
int  iFrAMA(
   string               symbol,            // nom du symbole
   ENUM_TIMEFRAMES      period,            // période
   int                  ma_period,         // période de la prise de moyenne
   int                  ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_APPLIED_PRICE   applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique des prix.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iFrAMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iFrAMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
```

© 2000-2015, MetaQuotes Software Corp.

```
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iFrAMA
#property indicator_label1  "iFrAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFrAMA,            // utiliser iFrAMA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iFrAMA;            // le type de la fonction
input int                  ma_period=14;                // la période de la prise de
input int                  ma_shift=0;                  // le décalage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;   // le type du prix
input string               symbol=" ";                  // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // le temps trame
//--- le tampon d'indicateur
double         iFrAMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iFrAMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Fractal Adaptive Moving Averag
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iFrAMABuffer,INDICATOR_DATA);
//--- spécifions le décalage
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
```

© 2000-2015, MetaQuotes Software Corp.

```
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iFrAMA)
      handle=iFrAMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[3];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du prix
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      //--- le type du prix
      handle=IndicatorCreate(name,period,IND_FRAMA,3,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iFrAMA pour la
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur iFrA
   short_name=StringFormat("iFrAMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2015, MetaQuotes Software Corp.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iFrAMA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iFrAMABuffer est plus grand, que les valeurs dans l'indicate
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iFrAMABuffer  par les valeurs de l'indicateur Fractal Ada
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iFrAMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur Fractal Adaptive Moving Averag
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Remplissons le tampon d'indicateur de l'indicateur iFrAMA        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Frac
                         int shift,         // le décalage
                         int ind_handle,    //le handle de l'indicateur iFrAMA
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iFrAMABuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iFrAMA, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iGatore

Rend le handle de l'indicateur Gator. L'Oscillateur montre la différence entre les lignes bleues et rouges de l'Alligator (l'histogramme supérieur) et la différence entre la ligne rouge et verte (l'histogramme inférieur).

```
int  iGator(
   string              symbol,            // nom du symbole
   ENUM_TIMEFRAMES     period,            // période
   int                 jaw_period,        // période pour le calcul des mâchoires
   int                 jaw_shift,         // décalage horizontal des mâchoires
   int                 teeth_period,      // période pour le calcul des dents
   int                 teeth_shift,       // décalage horizontal des dents
   int                 lips_period,       // période pour le calcul des lèvres
   int                 lips_shift,        // décalage horizontal des lèvres
   ENUM_MA_METHOD      ma_method,         // type de lissage
   ENUM_APPLIED_PRICE  applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

jaw_period

[in] La période de la prise de moyenne de la ligne bleue (la mâchoire d'alligator).

jaw_shift

[in] Le décalage de la ligne bleue de l'Alligator par rapport au graphique du prix. Il n'est pas directement raccordé avec le changement visuel de l'histogramme d'indicateur.

teeth_period

[in] La période de la prise de moyenne de la ligne rouge (les dents d'alligator).

teeth_shift

[in] Le décalage de la ligne rouge d'Alligator par rapport au graphique du prix. Il n'est pas directement raccordé avec le changement visuel de l'histogramme d'indicateur.

lips_period

[in] La période de la prise de moyenne de la ligne verte (les lèvres d'alligator).

lips_shift

[in] Le décalage de la ligne verte de l'Alligator par rapport au graphique du prix. Il n'est pas directement raccordé avec le changement visuel de l'histogramme d'indicateur.

ma_method

[in] La méthode de la prise de moyenne. Peut être l'une des valeurs ENUM_MA_METHOD.

© 2000-2015, MetaQuotes Software Corp.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - UPPER_HISTOGRAM, 1- le tampon coloré de l'histogramme supérieur, 2 - LOWER_HISTOGRAM, 3- le tampon coloré de l'histogramme inférieur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iGator.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iGator."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Gator Oscillator standard

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
//--- la construction GatorUp
#property indicator_label1  "GatorUp"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction GatorDown
#property indicator_label2  "GatorDown"
#property indicator_type2   DRAW_COLOR_HISTOGRAM
#property indicator_color2  clrGreen, clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
   Call_iGator,            // utiliser iGator
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iGator;       // le type de la fonction
input string               symbol=" ";             // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // le temps trame
input int                  jaw_period=13;          // la période pour la ligne des Mâc
input int                  jaw_shift=8;            // le décalage de la ligne des Mâch
input int                  teeth_period=8;         // la période pour la ligne des Der
input int                  teeth_shift=5;          // le décalage de la ligne des Mâch
input int                  lips_period=5;          // la période pour la ligne des Lèv
input int                  lips_shift=3;           // le décalage de la ligne des Lèvr
input ENUM_MA_METHOD       MA_method=MODE_SMMA;    // la méthode de la prise de moyenn
input ENUM_APPLIED_PRICE   applied_price=PRICE_MEDIAN;// le type de prix utilisé pour
//--- les tampons d'indicateur
double         GatorUpBuffer[];
double         GatorUpColors[];
double         GatorDownBuffer[];
double         GatorDownColors[];
//--- la variable pour stocker le handle de l'indicateur iGator
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- les valeurs des décalages pour l'histogramme supérieur et inférieur
int shift;
//--- stockerons le nombre de valeurs dans l'indicateur Gator Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,GatorUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,GatorUpColors,INDICATOR_COLOR_INDEX);
   SetIndexBuffer(2,GatorDownBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,GatorDownColors,INDICATOR_COLOR_INDEX);
/*
  Tous les décalages indiqués dans les paramètres se rapportent à l'indicateur Alligat
  C'est pourquoi les décalages indiqués ne produisent pas le décalage de l'indicateur
  dont les valeurs sont utilisées pour le calcul de Gator Oscillator!
*/
//--- Nous calculerons le décalage pour les histogrammes supérieurs et inférieurs, qui
   shift=MathMin(jaw_shift,teeth_shift);
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- malgré le fait qu'il y a deux histogrammes dans l'indicateur, on utilise le déca
   PlotIndexSetInteger(1,PLOT_SHIFT,shift);

//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iGator)
      handle=iGator(name,period,jaw_period,jaw_shift,teeth_period,teeth_shift,
                    lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[8];
      //--- les périodes et les décalages des lignes de l'Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
      //--- le type du lissage
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
      //--- le type du prix
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
      //--- créons le handle
      handle=IndicatorCreate(name,period,IND_GATOR,8,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iGator pour la
```

© 2000-2015, MetaQuotes Software Corp.

```
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Gator Oscillator a
   short_name=StringFormat("iGator(%s/%s, %d, %d ,%d, %d, %d, %d)",name,EnumToString(p
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iGator
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//---si c'est un premier lancement des calculs de notre indicateur ou a changé le nomb
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau GatorUpBuffer est plus grand, que les valeurs dans l'indicat
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
```

© 2000-2015, MetaQuotes Software Corp.

```
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur Gator Oscillator
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(GatorUpBuffer,GatorUpColors,GatorDownBuffer,GatorDownColo
      shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Gator Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iGator      |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &ups_buffer[],         // le tampon d'indicateur pou
                           double &up_color_buffer[],    // le tampon d'indicateur pou
                           double &downs_buffer[],       // le tampon d'indicateur pou
                           double &downs_color_buffer[], // le tampon d'indicateur pou
                           int u_shift,                  // le décalage pour l'histogr
                           int ind_handle,               // le handle de l'indicateur
                           int amount                    // le nombre de valeurs copié
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- emplissons la partie du tableau GatorUpBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,0,-u_shift,amount,ups_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iGator, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- emplissons la partie du tableau GatorUpColors par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,1,-u_shift,amount,up_color_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iGator, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- remplissons la partie du tableau GatorDownBuffer par les valeurs du tampon d'ind
   if(CopyBuffer(ind_handle,2,-u_shift,amount,downs_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iGator, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau GatorDownColors par les valeurs du tampon d'ind
   if(CopyBuffer(ind_handle,3,-u_shift,amount,downs_color_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iGator, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iIchimoku

Rend le handle de l'indicateur Ichimoku Kinko Hyo.

```
int  iIchimoku(
   string           symbol,           // nom du symbole
   ENUM_TIMEFRAMES  period,           // période
   int              tenkan_sen,       // période Tenkan-sen
   int              kijun_sen,        // période Kijun-sen
   int              senkou_span_b     // période Senkou Span B
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

tenkan_sen

[in] La période de la prise de moyenne Tenkan Sen.

kijun_sen

[in] La période de la prise de moyenne Kijun Sen.

senkou_span_b

[in] La période de la prise de moyenne Senkou Span B.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - TENKANSEN_LINE, 1 - KIJUNSEN_LINE, 2 - SENKOUSPANA_LINE, 3 - SENKOUSPANB_LINE, 4 - CHIKOUSPAN_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Demo_iIchimoku.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
```

© 2000-2015, MetaQuotes Software Corp.

```
#property description "des tampons d'indicateur pour l'indicateur technique iIchimoku.
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Ichimoku Kinko Hyo standa

#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   4
//--- la construction Tenkan_sen
#property indicator_label1  "Tenkan_sen"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Kijun_sen
#property indicator_label2  "Kijun_sen"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrBlue
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- la construction Senkou_Span
#property indicator_label3  "Senkou Span A;Senkou Span B" // dans Data Window deux cha
#property indicator_type3   DRAW_FILLING
#property indicator_color3  clrSandyBrown, clrThistle
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//--- la construction Chikou_Span
#property indicator_label4  "Chinkou_Span"
#property indicator_type4   DRAW_LINE
#property indicator_color4  clrLime
#property indicator_style4  STYLE_SOLID
#property indicator_width4  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iIchimoku,         // utiliser iIchimoku
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iIchimoku;       // le type de la fonction
input int                  tenkan_sen=9;              // la période Tenkan-sen
input int                  kijun_sen=26;              // la période Kijun-sen
input int                  senkou_span_b=52;          // la période Senkou Span B
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
double         Tenkan_sen_Buffer[];
double         Kijun_sen_Buffer[];
double         Senkou_Span_A_Buffer[];
double         Senkou_Span_B_Buffer[];
double         Chinkou_Span_Buffer[];
//--- la variable pour stocker le handle de l'indicateur iIchimoku
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Ichimoku Kinko Hyo
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,Tenkan_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(1,Kijun_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(2,Senkou_Span_A_Buffer,INDICATOR_DATA);
   SetIndexBuffer(3,Senkou_Span_B_Buffer,INDICATOR_DATA);
   SetIndexBuffer(4,Chinkou_Span_Buffer,INDICATOR_DATA);
//--- spécifions les décalages pour le canal Senkou Span sur kijun_sen bars au futur
   PlotIndexSetInteger(2,PLOT_SHIFT,kijun_sen);
//---il ne faut pas spécifier le décalage pour la ligne Chikou Span, parce que les dor
//--- sont stockées dans l'indicateur iIchimoku déjà avec le décalage
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iIchimoku)
      handle=iIchimoku(name,period,tenkan_sen,kijun_sen,senkou_span_b);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[3];
      //--- les périodes et les décalages des lignes de l'Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=tenkan_sen;
```

© 2000-2015, MetaQuotes Software Corp.

```
      pars[1].type=TYPE_INT;
      pars[1].integer_value=kijun_sen;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=senkou_span_b;
      //--- créons le handle
      handle=IndicatorCreate(name,period,IND_ICHIMOKU,3,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iIchimoku pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Ichi
   short_name=StringFormat("iIchimoku(%s/%s, %d, %d ,%d)",name,EnumToString(period),
                           tenkan_sen,kijun_sen,senkou_span_b);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iIchimoku
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau Tenkan_sen_Buffer est plus grand, que les valeurs dans l'ind
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur Ichimoku Kinko Hyo
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(Tenkan_sen_Buffer,Kijun_sen_Buffer,Senkou_Span_A_Buffer,S
      kijun_sen,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Ichimoku Kinko Hyo
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iIchimoku   |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &tenkan_sen_buffer[],     // le tampon d'indicateur
                           double &kijun_sen_buffer[],      // le tampon d'indicateur
                           double &senkou_span_A_buffer[],  // le tampon d'indicateur
                           double &senkou_span_B_buffer[],  // le tampon d'indicateur
                           double &chinkou_span_buffer[],   // le tampon d'indicateur
                           int senkou_span_shift,           // le décalage des lignes
                           int ind_handle,                  // le handle de l'indicate
                           int amount                       // le nombre de valeurs co
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau Tenkan_sen_Buffer par les valeurs du tampon d'i
   if(CopyBuffer(ind_handle,0,0,amount,tenkan_sen_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
```

© 2000-2015, MetaQuotes Software Corp.

```
      PrintFormat("1.On n'a pas réussi à copier les données de l'indicateur iGator, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau Kijun_sen_Buffer par les valeurs du tampon d'in
   if(CopyBuffer(ind_handle,1,0,amount,kijun_sen_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("2..On n'a pas réussi à copier les données de l'indicateur iGator, l
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau Chinkou_Span_Buffer par les valeurs du tampon c
//--- à senkou_span_shift>0 la ligne est décalée au futur sur senkou_span_shift bars
   if(CopyBuffer(ind_handle,2,-senkou_span_shift,amount,senkou_span_A_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("3.On n'a pas réussi à copier les données de l'indicateur iGator, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau Senkou_Span_A_Buffer par les valeurs du tampon
//--- à senkou_span_shift>0 la ligne est décalée au futur sur senkou_span_shift bars
   if(CopyBuffer(ind_handle,3,-senkou_span_shift,amount,senkou_span_B_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("4.On n'a pas réussi à copier les données de l'indicateur iGator, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau Senkou_Span_B_Buffer par les valeurs du tampon
//--- il ne faut pas prendre en considération le décalage pendant le copiage Chinkou S
//--- sont stockées dans l'indicateur iIchimoku déjà avec le décalage
   if(CopyBuffer(ind_handle,4,0,amount,chinkou_span_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("5.On n'a pas réussi à copier les données de l'indicateur iGator, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iBWMFI

Rend le handle de l'indicateur Market Facilitation Index. Seulement un tampon.

```
int  iBWMFI(
   string               symbol,             // nom du symbole
   ENUM_TIMEFRAMES      period,             // période
   ENUM_APPLIED_VOLUME  applied_volume      // type de volume pour le calcul
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

applied_volume

[in] Le volume utilisé. Peut être chacun des valeurs de l'énumération ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Demo_iBWMFI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iBWMFI."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- la construction iBWMFI
#property indicator_label1  "iBWMFI"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrLime,clrSaddleBrown,clrBlue,clrPink
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBWMFI,            // utiliser iBWMFI
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iBWMFI;          // le type de la fonction
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;// / le type du volume
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iBWMFIBuffer[];
double         iBWMFIColors[];
//--- la variable pour stocker le handle de l'indicateur iBWMFI
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Market Facilitation Index de B
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,iBWMFIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iBWMFIColors,INDICATOR_COLOR_INDEX);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iBWMFI)
      handle=iBWMFI(name,period,applied_volume);
   else
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      //--- le type du volume
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_BWMFI,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iIchimoku pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Mar
   short_name=StringFormat("iBWMFI(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iBWMFI
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau Tenkan_sen_Buffer est plus grand, que les valeurs dans l'ind
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- emplissons les tableaux par les valeurs de l'indicateur Market Facilitation Inde
//--- si FillArraysFromBuffer a rendu false, signifie que les données ne sont pas prêt
   if(!FillArraysFromBuffers(iBWMFIBuffer,iBWMFIColors,handle,values_to_copy)) return
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Market Facilitation Index de
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iBWMFI      |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &values[],    // le tampon d'indicateur des valeurs
                           double &colors[],    // le tampon d'indicateur des couleurs
                           int ind_handle,      // le handle de l'indicateur iBWMFI
                           int amount           // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iBWMFIBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iBWMFI, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- remplissons la partie du tableau iBWMFIColors par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,1,0,amount,colors)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iBWMFI, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iMomentum

Rend le handle de l'indicateur Momentum. Seulement un tampon.

```
int  iMomentum(
   string                symbol,            // nom du symbole
   ENUM_TIMEFRAMES       period,            // période
   int                   mom_period,        // période de la prise de moyenne
   ENUM_APPLIED_PRICE    applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

mom_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

applied_price

[in] Le prix utilisé. Peut être chacune des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Demo_iMomentum.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iMomentum."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul"
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre"
#property description "Tous les autres paramètres comme dans Momentum standard."

#property indicator_separate_window
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iMomentum
#property indicator_label1  "iMomentum"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMomentum,          // utiliser iMomentum
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iMomentum;       // le type de la fonction
input int                  mom_period=14;             // la période du Momentum
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iMomentumBuffer[];
//--- la variable pour stocker le handle de l'indicateur iMomentum
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Momentum
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iMomentumBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
```

© 2000-2015, MetaQuotes Software Corp.

```
     }
//--- créons le handle de l'indicateur
   if(type==Call_iMomentum)
      handle=iMomentum(name,period,mom_period,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=mom_period;
      //--- le type du prix
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MOMENTUM,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iMomentum pour
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Mome
   short_name=StringFormat("iMomentum(%s/%s, %d, %s)",name,EnumToString(period),
                           mom_period, EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iMomentum
```

© 2000-2015, MetaQuotes Software Corp.

```
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iMomentumBuffer est plus grand, que les valeurs dans l'indic
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iMomentumBuffer par les valeurs de l'indicateur Momentum
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iMomentumBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- etiendrons le nombre de valeurs dans l'indicateur Momentum
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iMomentum     |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Mome
                         int ind_handle,    // le handle de l'indicateur iMomentum
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iMomentumBuffer par les valeurs du tampon d'ind
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iMomentum, l
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iMFI

La calcul Money Flow Index.

```
int  iMFI(
   string                symbol,              // nom du symbole
   ENUM_TIMEFRAMES       period,              // période
   int                   ma_period,           // période de la prise de moyenne
   ENUM_APPLIED_VOLUME   applied_volume       // type de volume pour le calcul
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

applied_volume

[in] Le volume utilisé. Peut être l'un des ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                    Demo_iMFI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iMFI."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Money Flow Index standard

#property indicator_separate_window
#property indicator_buffers 1
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_plots   1
//--- la construction iMFI
#property indicator_label1  "iMFI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  20
#property indicator_level2  80
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMFI,              // utiliser iMFI
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iMFI;               // le type de la fonction
input int                  ma_period=14;                 // la période
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // le type du volume
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- le tampon d'indicateur
double         iMFIBuffer[];
//--- la variable pour stocker le handle de l'indicateur iMFI
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Money Flow Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iMFIBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iMFI)
      handle=iMFI(name,period,ma_period,applied_volume);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le type du volume
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_MFI,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iMFI pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame a été calculé l'indicateur Mone
   short_name=StringFormat("iMFI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period, EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//--- le nombre de valeurs copiées de l'indicateur iMFI
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iMFIBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iMFIBuffer par les valeurs de l'indicateur Money Flow Ind
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iMFIBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Money Flow Index
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iMFI          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // le tampon d'indicateur des valeurs Mone
                         int ind_handle,    // le handle de l'indicateur iMFI
                         int amount         // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
```

© 2000-2015, MetaQuotes Software Corp.

```
   ResetLastError();
//--- remplissons la partie du tableau iMFIBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iMFI, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iMA

Rend le handle de l'indicateur du moyen glissant. Seulement un tampon.

```
int  iMA(
   string                symbol,             // nom du symbole
   ENUM_TIMEFRAMES       period,             // période
   int                   ma_period,          // période de la prise de moyenne
   int                   ma_shift,           // décalage de l'indicateur à l'horizontale
   ENUM_MA_METHOD        ma_method,          // type de lissage
   ENUM_APPLIED_PRICE    applied_price       // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul du moyen glissant.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

ma_method

[in] La méthode de la prise de moyenne. Peut être l'une des valeurs ENUM_MA_METHOD.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
```

© 2000-2015, MetaQuotes Software Corp.

```
#property description "des tampons d'indicateur pour l'indicateur technique iMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Moving Average standard."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iMA
#property indicator_label1  "iMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMA,               // utiliser iMA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iMA;               // le type de la fonction
input int                  ma_period=10;                // la période de la moyenne
input int                  ma_shift=0;                  // le décalage
input ENUM_MA_METHOD       ma_method=MODE_SMA;          // le type du lissage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;   // le type du prix
input string               symbol=" ";                  // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // le temps trame
//--- le tampon d'indicateur
double         iMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//---stockerons le nombre de valeurs dans l'indicateur Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iMABuffer,INDICATOR_DATA);
//--- spécifions le décalage
```

© 2000-2015, MetaQuotes Software Corp.

```
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iMA)
      handle=iMA(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du lissage
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MA,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iMA pour la pai
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Moving Average a é
   short_name=StringFormat("iMA(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period, ma_shift,EnumToString(ma_method),EnumToString(ap
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iMA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- - si c'est un premier lancement des calculs de notre indicateur ou a changé le r
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iMABuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iMABuffer par les valeurs de l'indicateur Moving Average
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Moving Average
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iMA           |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // le tampon d'indicateur des valeurs Mov
                         int shift,          // le décalage
                         int ind_handle,     // le handle de l'indicateur iMA
                         int amount          // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iMABuffer par les valeurs du tampon d'indicateu
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iMA, le code
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iOsMA

Rend le handle de l'indicateur Moving Average of Oscillator. L'oscillateur OsMA montre la différence entre les valeurs MACD et sa ligne d'alarme. Seulement un tampon.

```
int  iOsMA(
   string               symbol,              // nom du symbole
   ENUM_TIMEFRAMES      period,              // période
   int                  fast_ema_period,     // période de la moyenne rapide
   int                  slow_ema_period,     // période de la moyenne lente
   int                  signal_period,       // période de la prise de moyenne de la di
   ENUM_APPLIED_PRICE   applied_price        // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

fast_ema_period

[in] La période de la prise de moyenne pour le calcul de la moyenne rapide glissant.

slow_ema_period

[in] La période de la prise de moyenne pour le calcul de la moyenne lente glissant.

signal_period

[in] La période de la prise de moyenne pour le calcul de la ligne d'alarme.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Dans certains systèmes cet oscillateur s'appelle l'histogramme MACD.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iOsMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iOsMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description ""Tous les autres paramètres comme dans Moving Average of Oscill

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iOsMA
#property indicator_label1  "iOsMA"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOsMA,             // utiliser iOsMA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iOsMA;           // le type de la fonction
input int                  fast_ema_period=12;        // la période de la moyenne rapi
input int                  slow_ema_period=26;        // la période de la moyenne lent
input int                  signal_period=9;           // la période de la prise de moy
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iOsMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iAMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Moving Average of Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iOsMABuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iOsMA)
      handle=iOsMA(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période rapide
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- la période lente
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- la période de la prise de moyenne de la différence entre les moyennes rapi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_OSMA,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iOsMA pour la p
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Moving Average of
   short_name=StringFormat("iOsMA(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
```

© 2000-2015, MetaQuotes Software Corp.

```
                              fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iOsMA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iOsMABuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iOsMA
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iOsMABuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

© 2000-2015, MetaQuotes Software Corp.

```
                              short_name,
                              values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur  Moving Average of Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iOsMA         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // le tampon d'indicateur des valeurs
                         int ind_handle,        // le handle de l'indicateur iOsMA
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iOsMABuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,0,amount,ama_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iOsMA, le co
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iMACD

Rend le handle de l'indicateur Moving Averages Convergence/Divergence. Dans ces systèmes, où on appelle OsMA l'histogramme MACD, cet indicateur est représenté en forme de deux lignes. Dans le terminal de client la convergence/divergence des moyens glissant se dessine en forme de l'histogramme.

```
int  iMACD(
   string               symbol,              // nom du symbole
   ENUM_TIMEFRAMES      period,              // période
   int                  fast_ema_period,     // période de la moyenne rapide
   int                  slow_ema_period,     // période de la moyenne lente
   int                  signal_period,       // période de la prise de moyenne de la di
   ENUM_APPLIED_PRICE   applied_price        // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

fast_ema_period

[in] La période de la prise de moyenne pour le calcul de la moyenne rapide glissant.

slow_ema_period

[in] La période de la prise de moyenne pour le calcul de la moyenne lente glissant.

signal_period

[in] La période de la prise de moyenne pour le calcul de la ligne d'alarme.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des bufers : 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iMACD.mq5 |
```

© 2000-2015, MetaQuotes Software Corp.

```
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iMACD."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans MACD standard."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- la construction MACD
#property indicator_label1  "MACD"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_DOT
#property indicator_width2  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMACD,             // utiliser iMACD
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iMACD;           // le type de la fonction
input int                  fast_ema_period=12;        // la période de la moyenne rapi
input int                  slow_ema_period=26;        // la période de la moyenne lent
input int                  signal_period=9;           // la période de la prise de moy
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- les tampons d'indicateur
double         MACDBuffer[];
double         SignalBuffer[];
//--- la variable pour stocker le handle de l'indicateur iMACD
int    handle;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Moving Averages Convergence/Di
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,MACDBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iMACD)
      handle=iMACD(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période rapide
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- la période lente
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- la période de la prise de moyenne de la différence entre les moyennes rapi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MACD,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iMACD pour la p
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Moving Averages Co
   short_name=StringFormat("iMACD(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
                           fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iMACD
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nom
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau MACDBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iMACD
//--- si FillArraysFromBuffers a rendu false, cela signifie que les données ne sont pa
   if(!FillArraysFromBuffers(MACDBuffer,SignalBuffer,handle,values_to_copy)) return(0)
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur  Moving Averages Convergence/D
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iMACD       |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &macd_buffer[],    // le tampon d'indicateur des val
                           double &signal_buffer[],  // le tampon d'indicateur de la l
                           int ind_handle,           // le handle de l'indicateur iMAC
                           int amount                // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iMACDBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,0,amount,macd_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iMACD, le co
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }

//--- remplissons la partie du tableau SignalBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iMACD, le co
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iOBV

Rend le handle de l'indicateur On Balance Volume. Seulement un tampon.

```
int  iOBV(
   string                symbol,             // nom du symbole
   ENUM_TIMEFRAMES       period,             // période
   ENUM_APPLIED_VOLUME   applied_volume      // type de volume pour le calcul
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

applied_volume

[in] Le volume utilisé. Peut être chacune des valeurs de l'énumération ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iOBV.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iOBV."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---  iOBV
#property indicator_label1  "iOBV"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOBV ,             // utiliser iOBV
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iOBV;               // le type de la fonction
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // le type du volume
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- les tampons d'indicateur
double         iOBVBuffer[];
//--- la variable pour stocker le handle de l'indicateur iOBV
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur On Balance Volume
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iOBVBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iOBV)
      handle=iOBV(name,period,applied_volume);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
      MqlParam pars[1];
      //--- le type du volume
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_OBV,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iOBV pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame On Balance Volume a été calculé
   short_name=StringFormat("iOBV(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iOBV
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nom
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iOBVBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iOBV
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iOBVBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- etiendrons le nombre de valeurs dans l'indicateur On Balance Volume
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iOBV          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &obv_buffer[],  // le tampon d'indicateur des valeurs
                         int ind_handle,        // le handle de l'indicateur iOBV
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iOBVBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,obv_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iOBV, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iSAR

Rend le handle de l'indicateur Parabolic Stop and Reverse system. Seulement un tampon.

```
int  iSAR(
   string           symbol,      // nom du symbole
   ENUM_TIMEFRAMES  period,      // période
   double           step,        // le pas du surcroît de vitesse - l'accélération
   double           maximum      //le coefficient maximum du parcours après le prix
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

step

[in] L'augmentation de niveau de Stop est 0.02 d'habitude.

maximum

[in] Le niveau de Stop maximum est 0.2 d'habitude.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                    Demo_iSAR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iSAR."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Parabolic Stop and Revers

#property indicator_chart_window
#property indicator_buffers 1
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_plots   1
//--- la construction iSAR
#property indicator_label1  "iSAR"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iSAR,              // utiliser iSAR
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iSAR;              // le type de la fonction
input double               step=0.02;                   // le pas - c'est le facteur
input double               maximum=0.2;                 // la valeur maximale du pas
input string               symbol=" ";                  // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // le temps trame
//--- les tampons d'indicateur
double         iSARBuffer[];
//--- la variable pour stocker le handle de l'indicateur iSAR
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Parabolic SAR
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iSARBuffer,INDICATOR_DATA);
//--- définissons pour la propriété PLOT_ARROW le code du symbole de l'ensemble Wingdi
   PlotIndexSetInteger(0,PLOT_ARROW,159);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iSAR)
      handle=iSAR(name,period,step,maximum);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la valeur du pas
      pars[0].type=TYPE_DOUBLE;
      pars[0].double_value=step;
      //--- la valeur limite du pas, qui peut être utilisée aux calculs
      pars[1].type=TYPE_DOUBLE;
      pars[1].double_value=maximum;
      handle=IndicatorCreate(name,period,IND_SAR,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iSAR pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Parabolic SAR a ét
   short_name=StringFormat("iSAR(%s/%s, %G, %G)",name,EnumToString(period),
                           step,maximum);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- le nombre de valeurs copiées de l'indicateur iSAR
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iSARBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iSAR
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iSARBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- stockerons le nombre de valeurs dans l'indicateur Parabolic SAR
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iSAR          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &sar_buffer[],  // le tampon d'indicateur des valeurs
                         int ind_handle,        // le handle de l'indicateur iSAR
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- remplissons la partie du tableau iSARBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,sar_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iSAR, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iRSI

Rend le handle de l'indicateur Relative Strength Index. Seulement un tampon.

```
int  iRSI(
   string               symbol,            // nom du symbole
   ENUM_TIMEFRAMES      period,            // période
   int                  ma_period,         // période de la prise de moyenne
   ENUM_APPLIED_PRICE   applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'index.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRSI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iRSI."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Relative Strength Index s

#property indicator_separate_window
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iRSI
#property indicator_label1  "iRSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- les limites pour l'affichage des valeurs dans la fenêtre de l'indicateur
#property indicator_maximum 100
#property indicator_minimum 0
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  70.0
#property indicator_level2  30.0
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRSI,              // utiliser iRSI
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iRSI;              // le type de la fonction
input int                  ma_period=14;                // la période de la prise de
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;   // le type du prix
input string               symbol=" ";                  // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // le temps trame
//--- le tampon d'indicateur
double         iRSIBuffer[];
//--- la variable pour stocker le handle de l'indicateur iRSI
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Relative Strength Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iRSIBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
```

© 2000-2015, MetaQuotes Software Corp.

```
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iRSI)
      handle=iRSI(name,period,ma_period,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la période de la moyenne
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- la valeur limite du pas, qui peut être utilisée aux calculs
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_RSI,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iRSI pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Relative Strength
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),
                           ma_period,applied_price);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

© 2000-2015, MetaQuotes Software Corp.

```
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iRSI
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iRSIBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur iRSI
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iRSIBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//---retiendrons le nombre de valeurs dans l'indicateur Relative Strength Index
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iRSI          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rsi_buffer[],  // le tampon d'indicateur des valeurs
                         int ind_handle,        // le handle de l'indicateur iSAR
```

© 2000-2015, MetaQuotes Software Corp.

```
                           int amount              // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iRSIBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,rsi_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iRSI, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iRVI

Rend le handle de l'indicateur Relative Vigor Index.

```
int  iRVI(
   string           symbol,        // nom du symbole
   ENUM_TIMEFRAMES  period,        // période
   int              ma_period      // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'index.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRVI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iRVI."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Relative Vigor Index star

#property indicator_separate_window
#property indicator_buffers 2
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_plots   2
//--- la construction RVI
#property indicator_label1  "RVI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRVI,              // utiliser iRVI
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iRVI;              // le type de la fonction
input int                  ma_period=10;                // la période des calculs
input string               symbol=" ";                  // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // le temps trame
//--- les tampons d'indicateur
double         RVIBuffer[];
double         SignalBuffer[];
//--- la variable pour stocker le handle de l'indicateur iRVI
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Relative Vigor Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,RVIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
```

© 2000-2015, MetaQuotes Software Corp.

```
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iRVI)
      handle=iRVI(name,period,ma_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      //--- la période pour les calculs
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_RVI,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iRVI pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Relative Vigor Inc
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
//--- le nombre de valeurs copiées de l'indicateur iRVI
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau RVIBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iRVI
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(RVIBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Relative Vigor Index
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iRVI        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rvi_buffer[],     // le tampon d'indicateur des valeu
                         double &signal_buffer[],  // le tampon d'indicateur de la lig
                         int ind_handle,           // le handle de l'indicateur iRVI
                         int amount                // le nombre de valeurs copiées
                         )
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iRVIBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,rvi_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iRVI, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- remplissons la partie du tableau SignalBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iRVI, le coc
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iStdDev

Rend le handle de l'indicateur Standard Deviation. Seulement un tampon.

```
int  iStdDev(
   string               symbol,            // nom du symbole
   ENUM_TIMEFRAMES      period,            // période
   int                  ma_period,         // période de la prise de moyenne
   int                  ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_MA_METHOD       ma_method,         // type de lissage
   ENUM_APPLIED_PRICE   applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom du symbole de l'instrument, dont les données doivent être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période de la prise de moyenne pour le calcul de l'indicateur.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

ma_method

[in] La méthode de la prise de moyenne. Peut être l'une des valeurs ENUM_MA_METHOD.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iStdDev.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
```

© 2000-2015, MetaQuotes Software Corp.

```
#property description "des tampons d'indicateur pour l'indicateur technique iStdDev."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Standard Deviation standa

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iStdDev
#property indicator_label1  "iStdDev"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStdDev,           // utiliser iStdDev
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iStdDev;         // le type de la fonction
input int                  ma_period=20;              // la période de la prise de moy
input int                  ma_shift=0;                // le décalage
input ENUM_MA_METHOD       ma_method=MODE_SMA;        // le type du lissage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iStdDevBuffer[];
//--- la variable pour stocker le handle de l'indicateur iStdDev
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Standard Deviation
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iStdDevBuffer,INDICATOR_DATA);
//--- spécifions le décalage
```

© 2000-2015, MetaQuotes Software Corp.

```
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iStdDev)
      handle=iStdDev(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du lissage
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_STDDEV,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iStdDev pour la
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Standard Deviation
   short_name=StringFormat("iStdDev(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(ma_method),EnumToString(app
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iStdDev
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iStdDevBuffer est plus grand, que les valeurs dans l'indicat
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Standard Deviation
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iStdDevBuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Standard Deviation
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iStdDev       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &std_buffer[],  // le tampon d'indicateur de la ligne
                         int std_shift,         // le décalage de la ligne Standard De
                         int ind_handle,        // le handle de l'indicateur iStdDev
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iStdDevBuffer par les valeurs du tampon d'indic
   if(CopyBuffer(ind_handle,0,-std_shift,amount,std_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iStdDev, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iStochastic

Rend le handle de l'indicateur Stochastic Oscillator.

```
int  iStochastic(
   string           symbol,          // nom du symbole
   ENUM_TIMEFRAMES  period,          // période
   int              Kperiod,         // K-période (le nombre de barres pour le calcul)
   int              Dperiod,         // D-période (la période de premier lissage)
   int              slowing,         // lissage final
   ENUM_MA_METHOD   ma_method,       // type de lissage
   ENUM_STO_PRICE   price_field      // méthode de calcul stochastique
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

Kperiod

[in] La K-période (le nombre de barres) pour le calcul de la ligne %K.

Dperiod

[in] La période de la prise de moyenne pour le calcul de la ligne %D.

slowing

[in] La valeur du ralentissement.

ma_method

[in] La méthode de la prise de moyenne. Peut être l'une des valeurs ENUM_MA_METHOD.

price_field

[in] Le paramètre du choix des prix pour le calcul. Peut être une des valeurs ENUM_STO_PRICE.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Note

Les numéros des tampons: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Demo_iStochastic.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
```

© 2000-2015, MetaQuotes Software Corp.

```
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iStochasti
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Stochastic Oscillator sta

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- la construction Stochastic
#property indicator_label1  "Stochastic"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- la construction Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- spécifions les valeurs de frontière de l'indicateur
#property indicator_minimum 0
#property indicator_maximum 100
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStochastic,       // utiliser iStochastic
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iStochastic;     // le type de la fonction
input int                  Kperiod=5;                 // la période K (le nombre de ba
input int                  Dperiod=3;                 // la période D (la période du l
input int                  slowing=3;                 // la période du lissage définit
input ENUM_MA_METHOD       ma_method=MODE_SMA;        // le type du lissage
input ENUM_STO_PRICE       price_field=STO_LOWHIGH;   // la méthode de calcul du stoch
input string               symbol=" ";                // le symbole
```

© 2000-2015, MetaQuotes Software Corp.

```
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- les tampons d'indicateur
double         StochasticBuffer[];
double         SignalBuffer[];
//--- la variable pour stocker le handle de l'indicateur iStochastic
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Stochastic Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement des tableaux aux tampons d'indicateur
   SetIndexBuffer(0,StochasticBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iStochastic)
      handle=iStochastic(name,period,Kperiod,Dperiod,slowing,ma_method,price_field);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[5];
      //--- la période K pour les calculs
      pars[0].type=TYPE_INT;
      pars[0].integer_value=Kperiod;
      //--- la période D pour le lissage primaire
      pars[1].type=TYPE_INT;
      pars[1].integer_value=Dperiod;
      //--- la période K pour le lissage définitif
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slowing;
      //--- le type du lissage
      pars[3].type=TYPE_INT;
```

© 2000-2015, MetaQuotes Software Corp.

```
      pars[3].integer_value=ma_method;
      //--- la méthode de calcul du stochastique
      pars[4].type=TYPE_INT;
      pars[4].integer_value=price_field;
      handle=IndicatorCreate(name,period,IND_STOCHASTIC,5,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iStochastic pou
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Stochastic Oscilla
   short_name=StringFormat("iStochastic(%s/%s, %d, %d, %d, %s, %s)",name,EnumToString
                           Kperiod,Dperiod,slowing,EnumToString(ma_method),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iStochastic
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau StochasticBuffer est plus grand, que les valeurs dans l'indi
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iStochastic
//--- si FillArraysFromBuffers a rendu false, cela signifie que les données ne sont pa
   if(!FillArraysFromBuffers(StochasticBuffer,SignalBuffer,handle,values_to_copy)) ret
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Stochastic Oscillator
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iStochastic |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &main_buffer[],    // le tampon d'indicateur des val
                           double &signal_buffer[],  // le tampon d'indicateur de la l
                           int ind_handle,           // le handle de l'indicateur iSto
                           int amount                // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau StochasticBuffer par les valeurs du tampon d'in
   if(CopyBuffer(ind_handle,MAIN_LINE,0,amount,main_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iStochastic,
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- remplissons la partie du tableau SignalBuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,SIGNAL_LINE,0,amount,signal_buffer)<0)
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
       PrintFormat("On n'a pas réussi à copier les données de l'indicateur iStochastic,
       //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
       return(false);
      }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iTEMA

Rend le handle de l'indicateur Triple Exponential Moving Average. Seulement un tampon.

```
int  iTEMA(
   string               symbol,            // nom du symbole
   ENUM_TIMEFRAMES      period,            // période
   int                  ma_period,         // période de la prise de moyenne
   int                  ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_APPLIED_PRICE   applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iTEMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iTEMA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
```

© 2000-2015, MetaQuotes Software Corp.

```
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Triple Exponential Moving

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iTEMA
#property indicator_label1  "iTEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTEMA,             // utiliser iTEMA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iTEMA;           // le type de la fonction
input int                  ma_period=14;              // la période de la prise de moy
input int                  ma_shift=0;                // le décalage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iTEMABuffer[];
//--- la variable pour stocker le handle de l'indicateur iTEMA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Triple Exponential Moving Aver
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iTEMABuffer,INDICATOR_DATA);
//--- spécifions le décalage
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
```

© 2000-2015, MetaQuotes Software Corp.

```
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iTEMA)
      handle=iTEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[3];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le décalage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- le type du prix
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TEMA,3,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iTEMA pour la p
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Triple Exponential
   short_name=StringFormat("iTEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2015, MetaQuotes Software Corp.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iTEMA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iTEMABuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Triple Exponential Moving
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iTEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- -retiendrons le nombre de valeurs dans l'indicateur Triple Exponential Moving Av
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
//| Remplissons le tampon d'indicateur de l'indicateur iTEMA         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &tema_buffer[], // le tampon d'indicateur des valeurs
                         int t_shift,           // le décalage de la ligne
                         int ind_handle,        // le handle de l'indicateur iTEMA
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iTEMABuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,-t_shift,amount,tema_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iTEMA, le co
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iTriX

Rend le handle de l'indicateur Triple Exponential Moving Averages Oscillator. Seulement un tampon.

```
int  iTriX(
   string               symbol,           // nom du symbole
   ENUM_TIMEFRAMES      period,           // période
   int                  ma_period,        // période de la prise de moyenne
   ENUM_APPLIED_PRICE   applied_price     // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

ma_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iTriX.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iTriX."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_plots   1
//--- la construction iTriX
#property indicator_label1  "iTriX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTriX,              // utiliser iTriX
   Call_IndicatorCreate     // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation              type=Call_iTriX;           // le type de la fonction
input int                   ma_period=14;              // la période
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // le type du prix
input string                symbol=" ";                // le symbole
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double          iTriXBuffer[];
//--- la variable pour stocker le handle de l'indicateur iTriX
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Triple Exponential Moving Aver
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iTriXBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- créons le handle de l'indicateur
   if(type==Call_iTriX)
      handle=iTriX(name,period,ma_period,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[2];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- le type du prix
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TRIX,2,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iTriX pour la p
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Triple Exponential
   short_name=StringFormat("iTriX(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iTriX
   int values_to_copy;
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iTriXBuffer est plus grand, que les valeurs dans l'indicateu
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Triple Exponential Moving
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iTriXBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Triple Exponential Moving Ave
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iTriX         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &trix_buffer[], // le tampon d'indicateur des valeurs
                         int ind_handle,        // le handle de l'indicateur iTriX
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iTriXBuffer par les valeurs du tampon d'indicat
   if(CopyBuffer(ind_handle,0,0,amount,trix_buffer)<0)
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
       PrintFormat("On n'a pas réussi à copier les données de l'indicateur iTriX, le co
       //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
       return(false);
      }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iWPR

Rend le handle de l'indicateur Larry Williams' Percent Range. Seulement un tampon.

```
int  iWPR(
   string           symbol,          // nom du symbole
   ENUM_TIMEFRAMES  period,          // période
   int              calc_period      // période de la prise de moyenne
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

calc_period

[in] La période (le nombre de barres) pour le calcul de l'indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iWPR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iWPR."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iWPR
#property indicator_label1  "iWPR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrCyan
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- spécifions les valeurs de frontière de l'indicateur
#property indicator_minimum -100
#property indicator_maximum 0
//--- les niveaux horizontaux dans la fenêtre de l'indicateur
#property indicator_level1  -20.0
#property indicator_level2  -80.0
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iWPR,              // utiliser iWPR
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iWPR;            // le type de la fonction
input int                  calc_period=14;            // la période
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iWPRBuffer[];
//--- la variable pour stocker le handle de l'indicateur iWPR
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Larry Williams' Percent Range
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iWPRBuffer,INDICATOR_DATA);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- créons le handle de l'indicateur
   if(type==Call_iWPR)
      handle=iWPR(name,period,calc_period);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      //--- la période
      pars[0].type=TYPE_INT;
      pars[0].integer_value=calc_period;
      handle=IndicatorCreate(name,period,IND_WPR,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iWPR pour la pa
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Williams' Percent
   short_name=StringFormat("iWPR(%s/%s, %d)",name,EnumToString(period),calc_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iWPR
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iWPRBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau iATRBuffer  par les valeurs de l'indicateur Williams' Per
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iWPRBuffer,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Williams' Percent Range
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iWPR          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &wpr_buffer[], // le tampon d'indicateur des valeurs W
                         int ind_handle,       // le handle de l'indicateur iWPR
                         int amount            // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iWPRBuffer par les valeurs du tampon d'indicate
   if(CopyBuffer(ind_handle,0,0,amount,wpr_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iWPR, le cod
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
```

© 2000-2015, MetaQuotes Software Corp.

```
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iVIDyA

Rend le handle de l'indicateur Variable Index Dynamic Average. Seulement un tampon.

```
int  iVIDyA(
   string               symbol,            // nom du symbole
   ENUM_TIMEFRAMES      period,            // période
   int                  cmo_period,        // période Chande Momentum
   int                  ema_period,        // période du facteur de lissage
   int                  ma_shift,          // décalage de l'indicateur à l'horizontale
   ENUM_APPLIED_PRICE   applied_price      // type de prix ou le handle
   );
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

cmo_period

[in] La période (le nombre de barres) pour le calcul Chande Momentum Oscillator.

ema_period

[in] La période (le nombre de barres) EMA pour le calcul du facteur de lissage.

ma_shift

[in] Le décalage de l'indicateur par rapport au graphique de prix.

applied_price

[in] Le prix utilisé. Peut être l'une des constantes de prix ENUM_APPLIED_PRICE ou le handle d'un autre indicateur.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iVIDyA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
```

© 2000-2015, MetaQuotes Software Corp.

```
#property description "des tampons d'indicateur pour l'indicateur technique iVIDyA."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre
#property description "Tous les autres paramètres comme dans Variable Index Dynamic Av

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- la construction iVIDyA
#property indicator_label1  "iVIDyA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVIDyA,            // utiliser iVIDyA
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iVIDyA;          // le type de la fonction
input int                  cmo_period=15;             // la période Chande Momentum
input int                  ema_period=12;             // la période du facteur du liss
input int                  ma_shift=0;                // le décalage
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // le type du prix
input string               symbol=" ";                // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // le temps trame
//--- le tampon d'indicateur
double         iVIDyABuffer[];
//--- la variable pour stocker le handle de l'indicateur iVIDyA
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Variable Index Dynamic Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau au tampon d'indicateur
   SetIndexBuffer(0,iVIDyABuffer,INDICATOR_DATA);
//--- spécifions le décalage
```

© 2000-2015, MetaQuotes Software Corp.

```
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iVIDyA)
      handle=iVIDyA(name,period,cmo_period,ema_period,ma_shift,applied_price);
   else
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[4];
      //--- la période Chande Momentum
      pars[0].type=TYPE_INT;
      pars[0].integer_value=cmo_period;
      //--- la période du facteur du lissage
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ema_period;
      //--- le décalage
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_shift;
      //--- le type du prix
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_VIDYA,4,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iVIDyA pour la
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Triple Exponential
   short_name=StringFormat("iVIDyA(%s/%s, %d, %d, %d, %s)",name,EnumToString(period),
                           cmo_period,ema_period,ma_shift,EnumToString(applied_price))
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
```

© 2000-2015, MetaQuotes Software Corp.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iVIDyA
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iWPRBuffer est plus grand, que les valeurs dans l'indicateur
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons le tableau par les valeurs de l'indicateur Variable Index Dynamic Av
//--- si FillArrayFromBuffer a rendu false, cela signifie que les données ne sont pas
   if(!FillArrayFromBuffer(iVIDyABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Variable Index Dynamic Averag
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons le tampon d'indicateur de l'indicateur iVIDyA        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &vidya_buffer[],// le tampon d'indicateur des valeurs
                         int v_shift,           // le décalage de la ligne
                         int ind_handle,        // le handle de l'indicateur iVIDyA
                         int amount             // le nombre de valeurs copiées
                         )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iVIDyABuffer par les valeurs du tampon d'indica
   if(CopyBuffer(ind_handle,0,-v_shift,amount,vidya_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iVIDyA, le c
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

iVolumes

Rend le handle de l'indicateur reflétant les volumes. Seulement un tampon.

```
int  iVolumes(
   string                symbol,              // nom du symbole
   ENUM_TIMEFRAMES       period,              // période
   ENUM_APPLIED_VOLUME   applied_volume       // type de volume
   )
```

Paramètres

symbol

[in] Le nom symbolique de l'instrument dont les données devraient être utilisées pour calculer l'indicateur. NULL signifie le symbole actuel.

period

[in] La valeur de la période peut être une des valeurs de l'énumération ENUM_TIMEFRAMES, 0 signifit le temps trame courant.

applied_volume

[in] Le volume utilisé. Peut être chacune des valeurs de l'énumération ENUM_APPLIED_VOLUME.

La valeur rendue

Rend le handle de l'indicateur technique indiqué, en cas de l'échec rend INVALID_HANDLE. Pour la désallocation de la mémoire de l'ordinateur de l' indicateur non plus utilisé plus il y a la fonction IndicatorRelease(), auquelle est transmis le handle de cet indicateur.

Exemple:

```
//+------------------------------------------------------------------+
//|                                                Demo_iVolumes.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "L'indicateur montre comment il faut recevoir les données"
#property description "des tampons d'indicateur pour l'indicateur technique iVolumes."
#property description "Le symbole et le temps trame sur lequel l'indicateur est calcul
#property description "sont spécifiés par les paramètres symbol et period."
#property description "Le moyen de la création du handle est spécifié par le paramètre

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- la construction iVolumes
#property indicator_label1  "iVolumes"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
```

© 2000-2015, MetaQuotes Software Corp.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| L'énumération des moyens de la création du handle                |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVolumes,          // utiliser iVolumes
   Call_IndicatorCreate    // utiliser IndicatorCreate
  };
//--- les paramètres d'entrée
input Creation             type=Call_iVolumes;           // le type de la fonction
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // le type du volume
input string               symbol=" ";                   // le symbole
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // le temps trame
//--- les tampons d'indicateur
double         iVolumesBuffer[];
double         iVolumesColors[];
//--- la variable pour stocker le handle de l'indicateur iVolumes
int    handle;
//--- la variable pour le stockage
string name=symbol;
//---le nom de l'indicateur sur le graphique
string short_name;
//--- stockerons le nombre de valeurs dans l'indicateur Volumes
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- le rattachement du tableau aux tampons d'indicateur
   SetIndexBuffer(0,iVolumesBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iVolumesColors,INDICATOR_COLOR_INDEX);
//--- définissons avec le symbole sur lequel l'indicateur est construit
   name=symbol;
//--- supprimons les espaces de la gauche et de la droite
   StringTrimRight(name);
   StringTrimLeft(name);
//--- si la longueur de la chaîne name est nulle après cela
   if(StringLen(name)==0)
     {
      //--- prenons le symbole du graphique, où on a lancé l'indicateur
      name=_Symbol;
     }
//--- créons le handle de l'indicateur
   if(type==Call_iVolumes)
      handle=iVolumes(name,period,applied_volume);
   else
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      //--- remplirons la structure par les valeurs des paramètres de l'indicateur
      MqlParam pars[1];
      //--- le type du prix
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_VOLUMES,1,pars);
     }
//--- si on n'a pas réussi à créer le handle
   if(handle==INVALID_HANDLE)
     {
      //--- informons sur l'échec et déduisons le numéro de l'erreur
      PrintFormat("On n'a pas réussi à créer le handle de l'indicateur iVolumes pour l
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- le travail de l'indicateur est terminé avant terme
      return(INIT_FAILED);
     }
//--- montrons sur quelle paire le symbole/temps trame l'indicateur Volumes a été calc
   short_name=StringFormat("iVolumes(%s/%s, %s)",name,EnumToString(period),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- l'exécution normale de l'initialisation de l'indicateur
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- le nombre de valeurs copiées de l'indicateur iVolumes
   int values_to_copy;
//--- apprenons le nombre de valeurs calculées dans l'indicateur
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() a rendu %d, le code de l'erreur %d",calculated,Get
      return(0);
     }
//--- si c'est un premier lancement des calculs de notre indicateur ou a changé le nor
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- ou s'il est nécessaire de calculer l'indicateur pour deux ou plus de barres (cel
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- si le tableau iVolumesBuffer r est plus grand, que les valeurs dans l'indi
      //--- autrement copions moins que la taille des tampons d'indicateur
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- signifie que notre indicateur est calculé non pour la première fois et dès
      //--- pour le calcul a été ajouté pas plus d'une barre
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- remplissons les tableaux par les valeurs de l'indicateur iVolumes
//--- si FillArraysFromBuffers a rendu false, cela signifie que les données ne sont pa
   if(!FillArraysFromBuffers(iVolumesBuffer,iVolumesColors,handle,values_to_copy)) ret
//--- formerons le message
   string comm=StringFormat("%s ==>  a été mise à jour des valeurs de l'indicateur %s:
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- déduisons le message de service sur le graphique
   Comment(comm);
//--- retiendrons le nombre de valeurs dans l'indicateur Volumes
   bars_calculated=calculated;
//--- rendons la valeur prev_calculated pour un appel suivant
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Remplissons les tampons d'indicateur de l'indicateur iVolumes    |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &volume_buffer[],   // le tampon d'indicateur des va
                           double &color_buffer[],    // le tampon d'indicateur des co
                           int ind_handle,            // le handle de l'indicateur iVo
                           int amount                 // le nombre de valeurs copiées
                           )
  {
//--- oblitérons le code de l'erreur
   ResetLastError();
//--- remplissons la partie du tableau iVolumesBuffer par les valeurs du tampon d'indi
   if(CopyBuffer(ind_handle,0,0,amount,volume_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iVolumes, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- remplissons la partie du tableau iVolumesColors par les valeurs du tampon d'indi
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(CopyBuffer(ind_handle,1,0,amount,color_buffer)<0)
     {
      //--- si le copiage n'a pas réussi, annonçons le code de l'erreur
      PrintFormat("On n'a pas réussi à copier les données de l'indicateur iVolumes, le
      //--- terminerons avec le résultat nul - cela signifie que l'indicateur sera cor
      return(false);
     }
//--- tout a réussi
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- effaçons le graphique à la suppression de l'indicateur
   Comment("");
  }
```

© 2000-2015, MetaQuotes Software Corp.

Travail avec les résultats de l'optimisation

Les fonctions pour l'organisation du traitement personnel des résultats de l'optimisation dans le testeur des stratégies. Elles peuvent être demandées dans les agents du test, ainsi que localement dans les experts et les scripts.

Au lancement de l'expert dans le testeur des stratégies on peut créer le tableau personnel à la base des types simples ou des structures simples(ne contiennent pas les lignes, les objets de la classe ou les objets des tableaux dynamiques). On peut sauvegarder cet ensemble de données à l'aide de la fonctionFrameAdd() dans une structure spéciale appelée un cadre (la trame). Chaque agent peut envoyer une série des trames au terminal à l'optimisation de l'expert. Tous les trames reçus dans l'ordre de l'entrée des agents s'inscrivent dans la fichier *.MQD au dossier le répertoire_du terminal/ MQL5/Files/Tester selon le nom de l'expert. L'entrée de la trame au terminal de client de l'agent du test génère l'événementTesterPass.

Les trames peuvent être stockées dans la mémoire de l'ordinateur ainsi qu'au fichier avec le nom spécifié. Il n'y a pas de limite au nombre de trames de la langue MQL5.

Fonction	Action
FrameFirst	Convertit l'indicateur de la lecture des trames au début et oblitére un filtre établi.
FrameFilter	Définit le filtre de la lecture des trames et met l'indicateur au début
FrameNext	Lit la trame et déplace l'indicateur au suivant
FrameInputs	Reçoit les paramètres input, sur lesquels la trame a été formé
FrameAdd	Ajoute la trame avec les données
ParameterGetRange	Reçoit pour la variable inputl'information sur la gamme des valeurs et sur le pas du changement à l'optimisation de l'expert dans le testeur des stratégies
ParameterSetRange	Établit des règles de l'utilisation de la variable inputà l'optimisation de l'expert dans le testeur des stratégies: la valeur, le pas du changement, les valeurs initiale et finale

Voir aussi

La statistique du test , Information sur le programme MQL5 lancé

© 2000-2015, MetaQuotes Software Corp.

FrameFirst

Convertit l'indicateur de la lecture des trames de l'optimisation au début et oblitére un filtre établi.

```
bool  FrameFirst();
```

La valeur rendue

Rend true en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

FrameFilter

Définit le filtre de la lecture des trames et met l'indicateur au début

```
bool  FrameFilter(
   const string  name,          // nom public / label
   long          id             // id public
   );
```

La valeur rendue

Rend true en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

Si à titre du premier paramètre on transmet la chaîne vide, le filtre fonctionnera seulement selon un paramètre numérique, c'est-à-dire toutes les trames avec id indiqué seront examinées. Si la valeur du deuxième paramètre est égale à ULONG_MAX, seulement le filtre de texte fonctionne.

L'appel FrameFilter("", ULONG_MAX) est égal à l'appelFrameFirst(), c'est-à-dire est équivalent à l'absence du filtre.

© 2000-2015, MetaQuotes Software Corp.

FrameNext

Lit la trame courant et déplace l'indicateur au suivant. Il y a 2 variantes de la fonction.

1. L'appel avec la réception d'une valeur numérique

```
bool  FrameNext(
   ulong&   pass,      // le numéro du passage dans l'optimisation, sur lequel la tran
   string&  name,      // nom public / label
   long&    id,        // id public
   double&  value      // valeur
   );
```

2. L'appel avec la réception de toutes les données de la trame

```
bool  FrameNext(
   ulong&   pass,      // le numéro du passage dans l'optimisation, sur lequel la tran
   string&  name,      // nom public / label
   long&    id,        // id public
   double&  value,     // valeur
   void&    data[]     // le tableau de n'importe quel type
   );
```

Paramètres

pass

[out] Le numéro du passage à l'optimisation dans le testeur des stratégies.

name

[out] Le nom de l'identificateur.

id

[out] La valeur de l'identificateur.

value

[out] Une valeur numérique unique.

data

[out] Un tableau de n'importe quel type.

La valeur rendue

Rend true en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

A l'utilisation de la deuxième variante de l'appel il est nécessaire de traiter correctement les données reçues au tableau*data[]*.

© 2000-2015, MetaQuotes Software Corp.

FrameInputs

Reçoitles paramètres input, sur lesquels la trame avec le numéro spécifiée du passage a été formée.

```
bool  FrameInputs(
   ulong    pass,                // le numéro du passage dans l'optimisation
   string&  parameters[],        // le tableau de chaînes de l'aspect "parameterN=valu
   uint&    parameters_count     // le nombre total de paramètres
   );
```

Paramètres

pass

[out] Le numéro du passage à l'optimisation dans le testeur des stratégies.

parameters

[out] Un tableau de chaînes décrivant les noms et les valeurs des paramètres

parameters_count

[out] Le nombre d'éléments dans le tableau *parameters[]*.

La valeur rendue

Rend true en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

Note

Ayant reçu la quantité de chaînes*parameters_count* dans le tableau*parameters[]*, on peut organiser le cycle pour le balayage de toutes les inscriptions. Cela permettra d'apprendre la valeur des paramètres d'entrée de l'expert pour le numéro spécifié du passage.

© 2000-2015, MetaQuotes Software Corp.

FrameAdd

Ajoute la trame avec les données Il y a 2 variantes de la fonction.

1. L'ajout des données à partir d'un fichier

```
bool  FrameAdd(
   const string  name,        // nom public / label
   long          id,          // id public
   double        value,       // valeur
   const string  filename     // le nom du fichier avec les données
   );
```

2. L'ajout des données du tableau de n'importe quel type

```
bool  FrameAdd(
   const string  name,        // nom public / label
   long          id,          // id public
   double        value,       // valeur
   const void&   data[]       // le tableau de n'importe quel type
   );
```

Paramètres

name

[in] Le label public la trame. Peut être utilisé comme le filtre dans la fonction FrameFilter().

id

[in] L'identificateur public de la trame. Peut être utilisé comme le filtre dans la fonctionFrameFilter().

value

[in] La valeur numérique pour l'inscription dans la trame. Est destiné à la transmission du résultat unique du passage comme dans la fonction OnTester().

filename

[in] Le nom du fichier, qui contient des données pour l'ajout dans une trame. Le fichier doit être dans un dossierMQL5/Files.

data

[in] Le tableau de n'importe quel type pour l'inscription dans une trame est transmis selon la référence.

La valeur rendue

Rend true en cas du succès, autrement - false. Pour recevoir l'information sur l'erreur, il faut appeler la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

ParameterGetRange

Reçoit pour la variable input l'information sur la gamme des valeurs et sur le pas du changement à l'optimisation de l'expert dans le testeur des stratégies. Il y a 2 variantes de la fonction.

1. La réception de l'information pour le paramètre input du type entier

```
bool  ParameterGetRange(
   const string  name,          // le nom du paramètre (de la variable input)
   bool&         enable,        // l'optimisation du paramètre est autorisée
   long&         value,         // la valeur du paramètre
   long&         start,         //la valeur initiale
   long&         step,          // le pas du changement
   long&         stop           // la valeur finale
   );
```

2. La réception de l'information pour le paramètre input du type matériel

```
bool  ParameterGetRange(
   const string  name,          // le nom du paramètre (de la variable input)
   double&       enable,        // l'optimisation du paramètre est autorisée
   double&       value,         // la valeur du paramètre
   double&       start,         // la valeur initiale
   double&       step,          // le pas du changement
   double&       stop           // la valeur finale
   );
```

Paramètres

name

[in] L'identificateur de la variable input. Telles variables sont les paramètres extérieurs du programme, dont les valeurs on peut spécifier au lancement sur le graphique ou au test séparé.

enable

[out] Le signe de ce qu'on peut utiliser ce paramètre pour le balayge des valeurs en train de l'optimisation dans le testeur des stratégies.

value

[out] La valeur du paramètre.

start

[out] La valeur initiale du paramètre à l'optimisation.

step

[out] Le pas du changement du paramètre au balayage de ses valeurs.

stop

[out] La valeur finale du paramètre à l'optimisation.

La valeur rendue

Rend true en cas de l'exécution réussie, autrement - false. Pour recevoir l'information sur l'erreur,

© 2000-2015, MetaQuotes Software Corp.

utilisez la fonction GetLastError().

Note

La fonction peut être appelée seulement à partir de gestionnairesOnTesterInit(), OnTesterPass() etOnTesterDeinit(). Est destinée à la réception de la valeur et de la gamme du changement des paramètres d'entrée de l'expert en train de l'optimisation dans le testeur des stratégies.

A l'appel dans OnTesterInit() on peut utiliser l'information reçue pour la redéfinition de la règle du balayage de chaquevariable input à l'aide de la fonctionParameterSetRange(). Ainsi spécifier les nouvelles valeurs Start, Stop, Step et même exclure entièrement ce paramètre de l'optimisation malgré les réglages dans le testeur des stratégies. Cela permet de créer les scripts personnels pour la gestion de l'espace des paramètres d'entrée à l'optimisation, c'est-à-dire exclure de l'optimisation des paramètres en fonction des valeurs des paramètres clés de l'expert.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
#property description "L'expert pour la démonstration de la fonction ParameterGetRange
#property description "Il est nécessaire de lancer dans le testeur des stratégies en m
//--- input parameters
input int                  Input1=1;
input double               Input2=2.0;
input bool                 Input3=false;
input ENUM_DAY_OF_WEEK     Input4=SUNDAY;

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- l'expert est destiné seulement au travail dans le testeur des stratégies
   if(!MQL5InfoInteger(MQL5_OPTIMIZATION))
     {
      MessageBox("Il est nécessaire de lancer dans le testeur des stratégies en mode d
      //---terminons le travail de l'expert avant terme et retirons du graphique
      return(INIT_FAILED);
     }
//---l'achèvement réussi de l'initialisation
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| TesterInit function                                              |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- l'exemple pour le paramètre input du type long
   string name="Input1";
   bool enable;
   long par1,par1_start,par1_step,par1_stop;
   ParameterGetRange(name,enable,par1,par1_start,par1_step,par1_stop);
   Print("Le premier paramètre");
   PrintFormat("%s=%d  enable=%s  from %d to %d with step=%d",
               name,par1,(string)enable,par1_start,par1_stop,par1_step);
//--- l'exemple pour le paramètre input du type double
   name="Input2";
   double par2,par2_start,par2_step,par2_stop;
   ParameterGetRange(name,enable,par2,par2_start,par2_step,par2_stop);
   Print("Le deuxième paramètre le paramètre");
   PrintFormat("%s=%G  enable=%s  from %G to %G with step=%G",
               name,par2,(string)enable,par2_start,par2_stop,par2_step);

//--- l'exemple pour le paramètre input du type bool
   name="Input3";
   long par3,par3_start,par3_step,par3_stop;
   ParameterGetRange(name,enable,par3,par3_start,par3_step,par3_stop);
   Print("Le troisième paramètre");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,(string)par3,(string)enable,
               (string)par3_start,(string)par3_stop);
//--- l'exemple pour le paramètre input du type l'énumération
   name="Input4";
   long par4,par4_start,par4_step,par4_stop;
   ParameterGetRange(name,enable,par4,par4_start,par4_step,par4_stop);
   Print("Le quatrième paramètre");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,EnumToString((ENUM_DAY_OF_WEEK)par4),(string)enable,
               EnumToString((ENUM_DAY_OF_WEEK)par4_start),
               EnumToString((ENUM_DAY_OF_WEEK)par4_stop));
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| TesterDeinit function                                            |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- ce message sortira à la fin de l'optimisation
   Print(__FUNCTION__," Optimisation completed");
  }
```

© 2000-2015, MetaQuotes Software Corp.

ParameterSetRange

>Établit des règles de l'utilisation dela variable input à l'optimisation de l'expert dans le testeur des stratégies: la valeur, le pas du changement, les valeurs initiale et finale. Il y a 2 variantes de la fonction.

1. L'installation des valeurs pour le paramètre input du type entier

```
bool  ParameterSetRange(
   const string  name,          // le nom du paramètre (de la variable input)
   bool          enable,        // autoriser l'optimisation du paramètre
   long          value,         // la valeur du paramètre
   long          start,         // la valeur initiale
   long          step,          // le pas du changement
   long          stop           // la valeur finale
   );
```

2. L'installation des valeurs pour le paramètre input du type matériel

```
bool  ParameterSetRange(
   const string  name,          // le nom du paramètre (de la variable input)
   double        enable,        // autoriser l'optimisation du paramètre
   double        value,         // la valeur du paramètre
   double        start,         // la valeur initiale
   double        step,          // le pas du changement
   double        stop           // la valeur finale
   );
```

Paramètres

name

[in] L'identificateur de la variableinput ou sinput. Telles variables sont les paramètres extérieurs du programme, dont les valeurs on peut spécifier au lancement.

enable

[in] Autoriser ce paramètre pour le balayage des valeurs en train de l'optimisation dans le testeur des stratégies.

value

[in] La valeur du paramètre.

start

[in] La valeur initiale du paramètre à l'optimisation.

step

[in] Le pas du changement du paramètre au balayage de ses valeurs.

stop

[in] La valeur finale du paramètre à l'optimisation.

La valeur rendue

© 2000-2015, MetaQuotes Software Corp.

Rend true en cas de l'exécution réussie, autrement - false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

La fonction peut être appelée seulement à partir du gestionnaireOnTesterInit() au lancement de l'optimisation dans le testeur des stratégies. Est destinée à la définition de la gamme et du pas du changement du paramètre, y compris elle permet d'exclure entièrement ce paramètre de la procédure de l'optimisation, malgré les réglages dans le testeur des stratégies. Permet d'utiliser aussi dans l'optimisation même les variables annoncées avec le modificateur sinput.

La fonction ParameterSetRange() permet de créer les scripts personnels de l'optimisation de l'expert dans le testeur des stratégies en fonction des valeurs de ses paramètres clés, c'est-à-dire insérer ou exclure de l'optimisation les paramètres d'entrée nécessaires, ainsi que spécifier la gamme et le pas du changement demandées.

© 2000-2015, MetaQuotes Software Corp.

Le travail avec les événements

Les fonctions pour le travail avec les événements d'utilisateur et les événements du minuteur. Sauf ces fonctions il y a des fonctions spéciales pour le traitement des événements prédéterminés.

Fonction	Action
EventSetMillisecondTimer	Lance le générateur des événements de la minuterie d'une haute permission avec la période moins 1 seconde pour un graphique courant
EventSetTimer	Lance le générateur des événements du minuteur avec la périodicité indiquée pour le graphique courant
EventKillTimer	Arrête sur le graphique courant la génération des événements selon le minuteur
EventChartCustom	Génère l'événement d'utilisateur pour le graphique indiqué

Voir aussi

Les types des événements du graphique

© 2000-2015, MetaQuotes Software Corp.

EventSetMillisecondTimer

Indique au terminal de client que pour l'expert donné ou l'indicateur il est nécessaire de générer les événements dela minuterieavec la périodicité moins d'une seconde.

```
bool  EventSetMillisecondTimer(
   int  milliseconds      // le nombre de millisecondes
   );
```

Réglages

milliseconds

[in] Le nombre de millisecondes définissant la périodicité de l'apparition des événements de la minuterie.

La valeur rendue

Rend true en cas de l'exécution réussie, autrement rend false. Pour recevoir le codede l'erreur il faut appeler la fonction GetLastError().

Note

Cette fonction est destinée aux cas, quand il fautla minuterie d'une haute permission, c'est-à-dire il faut recevoir les événements de la minuterie plus souvent, qu'une fois par seconde. Si la minuterie simple avec une période plus de 1 seconde est assez pour vous, utilisezEventSetTimer().

Dans le testeur des stratégies on utilise l'intervalle minimal à 1000 millisecondes. Dans le cas total à la réduction de la période de la minuterie augmente le temps du test, puisque augmente le nombre d'appels du gestionnaire des événements de la minuterie. Au travail en mode du temps réel les événements de la minuterie sont générés non plus souvent que 1 fois à 10-16 millisecondes ce qui est dû aux contraintes matérielles.

D'habitude, cette fonction doit être appelée de la fonctionOnInit() ou dans le constructeurde la classe. Pour traiter les événements provenant de la minuterie, l'expert ou l'indicateur doit avoir la fonctionOnTimer().

Chaque expert et chaque indicateur travaille avec sa minuterie et reçoit les événements seulement de lui. A l'achèvement du travail du programme mql5 la minuterie est supprimée forcément, si elle était créé, mais n'était pas déconnectée par la fonction EventKillTimer().

Pour chaque programme peut être lancé pas plus d'une minuterie. Chaque programme MQL5 et chaque graphique a sa propre file d'attente d'événements, où sont ajoutés tous les nouveaux événements entrants. Si l'événementTimerse trouve déjà dans la file d'attente ou cet événement est en état du traitement, on ne met pas un nouvel événement Timer dans la file d'attente du programme mql5.

© 2000-2015, MetaQuotes Software Corp.

EventSetTimer

Indique au terminal de client que pour l'expert donné ou l'indicateur il est nécessaire de générer les événements de minuteur avec la périodicité indiquée.

```
bool  EventSetTimer(
   int  seconds      // nombre de secondes
   );
```

Paramètres

seconds

[in] Le nombre de secondes définissant la périodicité de l'apparition des événements de minuteur.

La valeur rendue

En cas de l'exécution fructueuse rend true, autrement false.Pour la réception du code de l'erreur il faut appeler la fonction GetLastError().

Note

D'habitude, cette fonction doit être appelé de la fonction OnInit() ou dans le constructeur de la classe. Pour traiter les événements venant de minuteur, l'expert ou l'indicateur doit avoir la fonction OnTimer().

Chaque expert et chaque indicateur travaille avec le minuteur et reçoit les événements seulement de lui. A l'achèvement du travail du programme mql5 le minuteur est supprimé forcément, s'il était créé, mais n'était pas débrancher par la fonction EventKillTimer().

Pour chaque programme pas plus qu'un minuteur peut être lancé. Chaque programme MQL5 et chaque graphique a sa propre file d'attente d'événements, où sont ajoutés tous les nouveaux événements entrants. Si l'événementTimerse trouve déjà dans la file d'attente ou cet événement est en état du traitement, on ne met pas un nouvel événement Timer dans la file d'attente du programme mql5.

© 2000-2015, MetaQuotes Software Corp.

EventKillTimer

Indique au terminal de client qu'il est nécessaire d'arrêter la génération des événements de minuteur pour l'expert donné ou l'indicateur.

```
void  EventKillTimer();
```

La valeur rendue

Il n'y a pas de valeur rendue.

Note

D'habitude, cette fonction doit être appelé de la fonction OnDeinit() dans le cas où en fonction OnInit() la fonction EventSetTimer() a été appelée. Ou doit être appelé du destructeur de la classe, si dans le constructeur de cette classe la fonction EventSetTimer() a été appelée.

Chaque expert et chaque indicateur travaille avec le minuteur et reçoit les événements seulement de lui. A l'achèvement du travail du programme mql5 le minuteur est supprimé forcément, s'il était créé, mais n'était pas débrancher par la fonction EventKillTimer().

© 2000-2015, MetaQuotes Software Corp.

EventChartCustom

Génère l'événement d'utilisateur pour le graphique indiqué.

```
bool  EventChartCustom(
   long    chart_id,            // identificateur du graphique-destinataire de l'événe
   ushort  custom_event_id,     // identificateur de l'événement
   long    lparam,              // paramètre du type long
   double  dparam,              // paramètre du type double
   string  sparam               // paramètre de chaîne de l'événement
   );
```

Paramètres

chart_id

[in] L'identificateur du graphique. 0 signifie le graphique courant.

custom_event_id

[in] L'identificateur de l'événement d'utilisateur. Cet identificateur est ajouté automatiquement à la valeur CHARTEVENT_CUSTOM et est transcrit au type entier.

lparam

[in] Le paramètre de l'événement du type long, transmis à la fonction OnChartEvent.

dparam

[in] Le paramètre de l'événement du type double, transmis à la fonction OnChartEvent.

sparam

[in] Le paramètre de l'événement du type string, transmis à la fonction OnChartEvent. Si la chaîne a la longueur plus que 63 symboles, la chaîne est tronquée

La valeur rendue

Rend true en cas de la position réussie de l'événement d'utilisateur au tour des événements du graphique - du destinataire de l'événement. En cas de l'erreur rend false, pour recevoir un code de l'erreur utilisez GetLastError().

Note

L'expert ou l'indicateur fixé au graphique indiqué, traite l'événement donné à l'aide de la fonction OnChartEvent(int event_id, long& lparam, double& dparam, string& sparam).

Pour chaque type de l'événement les paramètres d'entrée de la fonction OnChartEvent() ont les valeurs définies, qui sont nécessaires au traitement de cet événement. Dans le tableau sont énumérés les événements et les valeurs, qui sont transmises par les paramètres.

L'événement	La valeur du paramètre id	La valeur du paramètre lparam	La valeur du paramètre dparam	La valeur du paramètre sparam
L'événement de la pression du clavier	CHARTEVENT_KEYDOWN	le code de la touche appuyée	Le nombre de pressions du bouton générées	La valeur de ligne d'un masque de bit,

© 2000-2015, MetaQuotes Software Corp.

			au cours de sa rétention dans l'état appuyé	décrivant le statut des boutons du clavier
Les événements de la souris(si la propriété est définie pour le graphique CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	La coordonnée X	La coordonnée Y	La valeur de chaîne d'un masque de bits décrivant l'état de boutons de souris
L'événement de la création de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_CREATE=true)	CHARTEVENT_OBJECT_CREATE	—	—	Le nom de l'objet créé graphique
L'événement du changement des propriétés de l'objet par le dialogue des propriétés	CHARTEVENT_OBJECT_CHANGE	—	—	Le nom de l'objet créé graphique
L'événement de la suppression de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_DELETE=true)	CHARTEVENT_OBJECT_DELETE	—	—	Le nom de l'objet graphique supprimé
L'événement du clic de la souris sur le graphique	CHARTEVENT_CLICK	La coordonnée X	La coordonnée Y	—
L'événement du clic de la souris sur l'objet graphique	CHARTEVENT_OBJECT_CLICK	La coordonnée X	La coordonnée Y	Le nom d'un objet graphique sur lequel l'événement s'est produit
L'événement du déplacement de	CHARTEVENT_OBJECT_DRAG	—	—	Le nom de l'objet graphique

© 2000-2015, MetaQuotes Software Corp.

l'objet graphique à l'aide de la souris				déplacé
L'événement de la fin de l'édition du texte dans l'objet graphique "Le champ de l'entrée"	CHARTEVENT_OBJECT_ENDEDIT	–	–	Le nom de l'objet graphique "Le champ de l'entrée", où l'édition du texte s'est achevée
L'événement du changement du graphique	CHARTEVENT_CHART_CHANGE	–	–	–
L'événement d'utilisateur avec le numéro N	CHARTEVENT_CUSTOM+N	La valeur, spécifiée par la fonction EventChartCustom()	La valeur, spécifiée par la fonction EventChartCustom()	La valeur, spécifiée par la fonction EventChartCustom()

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//|                                              ButtonClickExpert.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

string buttonID="Button";
string labelID="Info";
int broadcastEventID=5000;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- créons le bouton, pour la transmission des événements d'utilisateur
   ObjectCreate(0,buttonID,OBJ_BUTTON,0,100,100);
   ObjectSetInteger(0,buttonID,OBJPROP_COLOR,clrWhite);
   ObjectSetInteger(0,buttonID,OBJPROP_BGCOLOR,clrGray);
   ObjectSetInteger(0,buttonID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_XSIZE,200);
   ObjectSetInteger(0,buttonID,OBJPROP_YSIZE,50);
   ObjectSetString(0,buttonID,OBJPROP_FONT,"Arial");
   ObjectSetString(0,buttonID,OBJPROP_TEXT,"Bouton");
   ObjectSetInteger(0,buttonID,OBJPROP_FONTSIZE,10);
   ObjectSetInteger(0,buttonID,OBJPROP_SELECTABLE,0);

//--- créons la marque pour afficher l'information
   ObjectCreate(0,labelID,OBJ_LABEL,0,100,100);
   ObjectSetInteger(0,labelID,OBJPROP_COLOR,clrRed);
   ObjectSetInteger(0,labelID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,labelID,OBJPROP_YDISTANCE,50);
   ObjectSetString(0,labelID,OBJPROP_FONT,"Trebuchet MS");
   ObjectSetString(0,labelID,OBJPROP_TEXT,"Pas d'information");
   ObjectSetInteger(0,labelID,OBJPROP_FONTSIZE,20);
   ObjectSetInteger(0,labelID,OBJPROP_SELECTABLE,0);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ObjectDelete(0,buttonID);
   ObjectDelete(0,labelID);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
```

© 2000-2015, MetaQuotes Software Corp.

```
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- contrôlerons l'événement sur la pression du bouton de la souris
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      string clickedChartObject=sparam;
      //--- si la pression sur l'objet avec le nom buttonID
      if(clickedChartObject==buttonID)
        {
         //--- Etat du bouton - on a appuyé le bouton ou non
         bool selected=ObjectGetInteger(0,buttonID,OBJPROP_STATE);
         //--- notez un message de réglage
         Print("Bouton est appuyé = ",selected);
         int customEventID; // numéro de l'événement d'utilisateur pour l'expédition
         string message;    // message pour l'expédition dans l'événement
         //--- si le bouton est appuyé
         if(selected)
           {
            message="bouton est appuyé";
            customEventID=CHARTEVENT_CUSTOM+1;
           }
         else // bouton n'est pas appuyé
           {
            message="Bouton est appuyé";
            customEventID=CHARTEVENT_CUSTOM+999;
           }
         //--- envoyons l'événement d'utilisateur au graphique
         EventChartCustom(0,customEventID-CHARTEVENT_CUSTOM,0,0,message);
         //--- envoyons le message aux tous les graphiques ouverts
         BroadcastEvent(ChartID(),0,"Broadcast Message");
         //--- message de réglage
         Print("On a expédié l'événement avec ID = ",customEventID);
        }
      ChartRedraw();// copions forcément tous les objets sur le graphique
     }

//--- vérifions l'événement sur l'appartenance aux événements d'utilisateur
   if(id>CHARTEVENT_CUSTOM)
     {
      if(id==broadcastEventID)
        {
         Print("Ont reçu le message de diffusion du graphique avec id = "+lparam);
        }
      else
        {
         //--- lirons le message de texte dans l'événement
         string info=sparam;
         Print("On traite l'événement D'UTILISATEUR avec  ID = ",id);
         //--- déduisons le message dans la marque
         ObjectSetString(0,labelID,OBJPROP_TEXT,sparam);
         ChartRedraw();// copions forcément tous les objets sur le graphique
        }
     }
  }
//+------------------------------------------------------------------+
//| envoyer le message de diffusion aux graphiques ouverts           |
//+------------------------------------------------------------------+
void BroadcastEvent(long lparam,double dparam,string sparam)
```

© 2000-2015, MetaQuotes Software Corp.

```
  {
   int eventID=broadcastEventID-CHARTEVENT_CUSTOM;
   long currChart=ChartFirst();
   int i=0;
   while(i<CHARTS_MAX)                // nous avons sans faute plus de CHARTS_MAX des
     {
      EventChartCustom(currChart,eventID,lparam,dparam,sparam);
      currChart=ChartNext(currChart); // sur la base du précédent nous recevrons un no
      if(currChart==-1) break;        // ont atteint la fin de la liste des graphiques
      i++;                            // n'oublierons pas d'augmenter le compteur
     }
  }
//+------------------------------------------------------------------+
```

Voir aussi

Les événements du terminal de client, la Fonction du traitement des événements

© 2000-2015, MetaQuotes Software Corp.

Le travail avec OpenCL

Les programmes surOpenCLsont destinés à l'exécution des calculs sur les cartes graphiques avec le soutien du standard OpenCL 1.1 ou plus haut. Les cartes graphiques modernes contiennent centaines de petits processeurs spécialisés qui peuvent simultanément effectuer des opérations mathématiques simples sur les threads entrants des données. Le langage OpenCL prend en charge l'organisation de tels calculs parallèles et permet d'obtenir l'accélération immense pour une certaine classe des tâches.

Les fonctions pour l'exécution des programmes sur OpenCL:

Fonction	Action
CLHandleType	Rend le type OpenCL du handle à titre de la valeur de l'énumération ENUM_OPENCL_HANDLE_TYPE
CLGetInfoInteger	Rend la valeur de la propriété entière pour l'objet OpenCL ou le dispositif
CLContextCreate	Crée le contexte OpenCL
CLContextFree	Supprime le contexte OpenCL
CLGetDeviceInfo	Reçoit la propriété du dispositif du pilote informatique OpenCL
CLProgramCreate	Crée le programme OpenCL du code initial
CLProgramFree	Supprime le programme OpenCL
CLKernelCreate	Crée la fonction du lancement de OpenCL
CLKernelFree	Supprime la fonction du lancement de OpenCL
CLSetKernelArg	Expose le paramètre de la fonction OpenCL
CLSetKernelArgMem	Expose le tampon OpenCL comme un paramètre de la fonction OpenCL
CLBufferCreate	Crée le tampon OpenCL
CLBufferFree	Supprime le tampon OpenCL
CLBufferWrite	Inscrit le tableau au tampon OpenCL
CLBufferRead	Lit le tampon OpenCL au tableau
CLExecute	Exécute le programme OpenCL

© 2000-2015, MetaQuotes Software Corp.

CLHandleType

Rend le type OpenCL du handle à titre de la valeur de l'énumération ENUM_OPENCL_HANDLE_TYPE.

```
ENUM_OPENCL_HANDLE_TYPE  CLHandleType(
   int  handle     // le handle de l'objet OpenCL
   );
```

Paramètres

handle

[in] Le handle sur l'objet OpenCL: le contexte, le noyau, le tampon ou le programme OpenCL.

La valeur rendue

Le type du handle OpenCL à titre de la valeur de l'énumération ENUM_OPENCL_HANDLE_TYPE.

ENUM_OPENCL_HANDLE_TYPE

Identificateur	La description
OPENCL_INVALID	Le handle incorrect
OPENCL_CONTEXT	Le handle du cotexte OpenCL
OPENCL_PROGRAM	Le handle du programme OpenCL
OPENCL_KERNEL	Le handle du noyau OpenCL
OPENCL_BUFFER	Le handle du tampon OpenCL

© 2000-2015, MetaQuotes Software Corp.

CLGetInfoInteger

Rend la valeur de la propriété entière pour l'objet OpenCL ou le dispositif

```
long  CLGetInfoInteger(
   int  handle,                          // le handle de l'objet OpenCL ou le numéro
   ENUM_OPENCL_PROPERTY_INTEGER  prop    // la propriété demandée
   );
```

Paramètres

handle

[in] Le handle sur l'objet OpenCL ou le numéro du dispositif OpenCL. Le numérotage OpenCL des dispositifs commence par le zéro

prop

[in] Le type de la propriété demandée de l'énumérationENUM_OPENCL_PROPERTY_INTEGER, dont la valeur il faut recevoir.

La valeur rendue

La valeur de la propriété indiquée à l'exécution réussie ou-1 en cas de l'erreur. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

ENUM_OPENCL_PROPERTY_INTEGER

Identificateur	La description	Type
CL_DEVICE_COUNT	Le nombre des dispositifs avec le soutien de OpenCL. Pour cette propriété on ne demande pas l'indication du premier paramètre, c'est-à-dire on peut transmettre la valeur nulle pour le paramètre*handle*.	int
CL_DEVICE_TYPE	Le type du dispositif	ENUM_CL_DEVICE_TYPE
CL_DEVICE_VENDOR_ID	L'identificateur unique du producteur	uint
CL_DEVICE_MAX_COMPUTE_UNITS	Le nombre de tâches parallèles calculées dans le dispositif OpenCL. Un groupe de travail accomplit une tâche calculatoire. La valeur minimale est 1	uint
CL_DEVICE_MAX_CLOCK_FREQUENCY	La fréquence maximale en MHZ établie du dispositif.	uint
CL_DEVICE_GLOBAL_MEM_SIZE	La taille de la mémoire globale en octets	ulong
CL_DEVICE_LOCAL_MEM_SIZE	La taille de la mémoire locale	uint

© 2000-2015, MetaQuotes Software Corp.

	des données traitées (de la scène) en octets	
CL_BUFFER_SIZE	Реальный размер буфера OpenCL в байтах	ulong

L'énumération ENUM_CL_DEVICE_TYPE contient les types possibles des dispositifs avec le soutien OpenCL. On peut recevoir le type du dispositif selon son numéro ou le handle de l'objet OpenCL à l'aide de l'appel CLGetInfoInteger(handle_or_deviceN, CL_DEVICE_TYPE).

ENUM_CL_DEVICE_TYPE

Identificateur	La description
CL_DEVICE_ACCELERATOR	L'accélérateur spécialisé OpenCL (par exemple, IBM CELL Blade).
CL_DEVICE_CPU	L'utilisation du processeur central de l'ordinateur à titre du dispositif OpenCL. Le processeur central peut avoir un ou plus de noyaux calculatoires.
CL_DEVICE_GPU	Le dispositif OpenCL basé sur la carte vidéo.
CL_DEVICE_DEFAULT	Le dispositif OpenCL par défaut. Le dispositif CL_DEVICE_TYPE_CUSTOM ne peut pas être le dispositif par défaut.
CL_DEVICE_CUSTOM	Les accélérateurs spécialisés qui ne soutiennt pas les programmes au OpenCL C.

Exemple:

```
void OnStart()
  {
   int cl_ctx;
//--- l'initialisation du contexte OpenCL
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL not found");
      return;
     }
//--- déduisons l'information totale sur le dispositif   OpenCL
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY)," MHz"
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE)," by
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE)," byte
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

CLGetInfoString

Rend la valeur de ligne pour l'objet OpenCL ou le dispositif.

```
bool  CLGetInfoString(
   int  handle,                          // le handle de l'objet OpenCL ou le numéro
   ENUM_OPENCL_PROPERTY_STRING  prop,    // la propriété demandée
   string&  value                        // la ligne selon la référence
   );
```

Réglages

handle

[in] Le handle sur l'objet OpenCL ou le numéro du dispositif OpenCL. Le numérotage OpenCL des dispositifs commence par le zéro.

prop

[in] Le type de la propriété demandée de l'énumération ENUM_OPENCL_PROPERTY_STRING, dont la valeur il faut recevoir.

value

[out] La ligne pour la réception de la valeur de la propriété.

La valeur rendue

En cas de l'exécution successive rend true, autrement rend false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

ENUM_OPENCL_PROPERTY_STRING

Identificateur	La description
CL_PLATFORM_PROFILE	CL_PLATFORM_PROFILE - Le profil OpenCL. Le nom du profil peut être une des valeurs: • FULL_PROFILE - la réalisation soutient OpenCL (la fonctionnalité est définie comme la partie de la spécification du noyau et ne demande pas les extensions supplémentaires pour le soutien d' OpenCL); • EMBEDDED_PROFILE - la réalisation soutient OpenCL en forme du complément. Le profil complété est défini comme la sous-multitude pour chaque version OpenCL.
CL_PLATFORM_VERSION	La version OpenCL
CL_PLATFORM_VENDOR	Le nom du fabricant de l'appareil
CL_PLATFORM_EXTENSIONS	La liste des extensions soutenus par la plateforme. Les noms des extensions doivent être soutenus par toutes les dispositifs liés avec cette plateforme
CL_DEVICE_NAME	Le nom du dispositif

© 2000-2015, MetaQuotes Software Corp.

CL_DEVICE_VENDOR	Le nom du fabricant
CL_DRIVER_VERSION	La version du pilote informatique OpenCL dans le format major_number.minor_number
CL_DEVICE_PROFILE	Le profil du dispositif OpenCL. Le nom du profil peut être une des valeurs: • FULL_PROFILE - la réalisation soutient OpenCL (la fonctionnalité est définie comme la partie de la spécification du noyau et ne demande pas les extensions supplémentaires pour le soutien d' OpenCL); • EMBEDDED_PROFILE - la réalisation soutient OpenCL en forme du complément. Le profil complété est défini comme la sous-multitude pour chaque version OpenCL.
CL_DEVICE_VERSION	La version OpenCL dans le format "OpenCL<space><major_version.minor_version><space><vendor-specific information>"
CL_DEVICE_EXTENSIONS	La liste des extensions soutenus par le dispositif. La liste peut insérer les extensions soutenus par le producteur et contenir un ou plus de noms approuvés: cl_khr_int64_base_atomics cl_khr_int64_extended_atomics cl_khr_fp16 cl_khr_gl_sharing cl_khr_gl_event cl_khr_d3d10_sharing cl_khr_dx9_media_sharing cl_khr_d3d11_sharing
CL_DEVICE_BUILT_IN_KERNELS	La liste des noyaux, soutenus par le dispositif, divisé par ";" .
CL_DEVICE_OPENCL_C_VERSION	La version maximale soutenue par le compilateur pour ce dispositif. Le format de la version: "OpenCL<space>C<space><major_version.minor_version><space><vendor-specific information> "

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
void OnStart()
  {
   int cl_ctx;
   string str;
//--- l'initialisation du contexte OpenCL
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL not found");
      return;
     }
//--- déduisons l'information sur la plateforme
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_NAME,str))
      Print("OpenCL platform name: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VENDOR,str))
      Print("OpenCL platform vendor: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VERSION,str))
      Print("OpenCL platform ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_PROFILE,str))
      Print("OpenCL platform profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_EXTENSIONS,str))
      Print("OpenCL platform ext: ",str);
//--- déduisons l'information sur le dispositif
   if(CLGetInfoString(cl_ctx,CL_DEVICE_NAME,str))
      Print("OpenCL device name: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_PROFILE,str))
      Print("OpenCL device profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_BUILT_IN_KERNELS,str))
      Print("OpenCL device kernels: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_EXTENSIONS,str))
      Print("OpenCL device ext: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VENDOR,str))
      Print("OpenCL device vendor: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VERSION,str))
      Print("OpenCL device ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_OPENCL_C_VERSION,str))
      Print("OpenCL open c ver: ",str);
//--- déduisons l'information totale sur le dispositif  OpenCL
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY));
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE));
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE));
//---
  }
```

© 2000-2015, MetaQuotes Software Corp.

CLContextCreate

Crée le contexte OpenCL et rend le handle sur lui.

```
int  CLContextCreate(
   int  device     // le numéro d'ordre du dispositif OpenCL ou la macro
   );
```

Paramètres

device

[in] Le numéro du dispositif OpenCL dans le système par ordre. Au lieu du numéro concret on peut indiquer une des valeurs: CL_USE_ANY - il est permis d'utiliser n'importe quelle installation accessible avec le soutien OpenCL; CL_USE_GPU_ONLY - l'émulation OpenCL est interdite et il est permis d'utiliser seulement les dispositifs spécialisées avec le soutien de OpenCL (la carte vidéo).

La valeur rendue

Le handle sur le contexte OpenCL à la création réussie, ou -1 en cas de l'erreur. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

CLContextFree

Supprime le contexte OpenCL.

```
void  CLContextFree(
   int  context     // le handle sur le cotexte OpenCL
   );
```

Paramètres

context

[in] Le handle du cotexte OpenCL.

La valeur rendue

Non. En cas de l'erreur intérieure change la valeur _LastError. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

CLGetDeviceInfo

Reçoit la propriété du dispositif du pilote informatique OpenCL.

```
bool  CLGetDeviceInfo(
   int     handle,          // le handle du dispositif OpenCL
   int     property_id,     // l'identificateur de la propriété demandée
   uchar&  data[],          // le tableau pour la réception des données
   uint&   size             // le décalage dans le tableau dans les éléments, par défa
   );
```

Paramètres

handle

[in] Le numéro du dispositif OpenCL ou le handle OpenCL, créé par la fonctionCLContextCreate().

property_id

[in] L'identificateur de la propriété, dont il est nécessaire de recevoir sur le dispositif OpenCL. Peut être une des valeurs prédéterminées énumérées au tableau plus bas.

data[]

[out] Le tableau pour la réception des données sur la propriété demandée.

size

[out] La taille des données reçues dans le tableau*data[]*.

La valeur rendue

En cas de l'exécution successive rend true, autrement rend false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

Pour les tableaux unidimentionnels le numéro de l'élément, par qui commence la lecture des données pour l'inscription au tampon OpenCL, est calculé en prenant en compte le drapeau AS_SERIES.

La liste des identificateurs admissibles des propriétés OpenCL du dispositif

On peut trouver la description exacte de la propriété et sa destination sur le site officiel OpenCL.

Identificateur	La valeur
CL_DEVICE_TYPE	0x1000
CL_DEVICE_VENDOR_ID	0x1001
CL_DEVICE_MAX_COMPUTE_UNITS	0x1002
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS	0x1003
CL_DEVICE_MAX_WORK_GROUP_SIZE	0x1004
CL_DEVICE_MAX_WORK_ITEM_SIZES	0x1005
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR	0x1006

© 2000-2015, MetaQuotes Software Corp.

CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT	0x1007
CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT	0x1008
CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG	0x1009
CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT	0x100A
CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE	0x100B
CL_DEVICE_MAX_CLOCK_FREQUENCY	0x100C
CL_DEVICE_ADDRESS_BITS	0x100D
CL_DEVICE_MAX_READ_IMAGE_ARGS	0x100E
CL_DEVICE_MAX_WRITE_IMAGE_ARGS	0x100F
CL_DEVICE_MAX_MEM_ALLOC_SIZE	0x1010
CL_DEVICE_IMAGE2D_MAX_WIDTH	0x1011
CL_DEVICE_IMAGE2D_MAX_HEIGHT	0x1012
CL_DEVICE_IMAGE3D_MAX_WIDTH	0x1013
CL_DEVICE_IMAGE3D_MAX_HEIGHT	0x1014
CL_DEVICE_IMAGE3D_MAX_DEPTH	0x1015
CL_DEVICE_IMAGE_SUPPORT	0x1016
CL_DEVICE_MAX_PARAMETER_SIZE	0x1017
CL_DEVICE_MAX_SAMPLERS	0x1018
CL_DEVICE_MEM_BASE_ADDR_ALIGN	0x1019
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE	0x101A
CL_DEVICE_SINGLE_FP_CONFIG	0x101B
CL_DEVICE_GLOBAL_MEM_CACHE_TYPE	0x101C
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE	0x101D
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE	0x101E
CL_DEVICE_GLOBAL_MEM_SIZE	0x101F
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE	0x1020
CL_DEVICE_MAX_CONSTANT_ARGS	0x1021
CL_DEVICE_LOCAL_MEM_TYPE	0x1022
CL_DEVICE_LOCAL_MEM_SIZE	0x1023
CL_DEVICE_ERROR_CORRECTION_SUPPORT	0x1024

© 2000-2015, MetaQuotes Software Corp.

CL_DEVICE_PROFILING_TIMER_RESOLUTION	0x1025
CL_DEVICE_ENDIAN_LITTLE	0x1026
CL_DEVICE_AVAILABLE	0x1027
CL_DEVICE_COMPILER_AVAILABLE	0x1028
CL_DEVICE_EXECUTION_CAPABILITIES	0x1029
CL_DEVICE_QUEUE_PROPERTIES	0x102A
CL_DEVICE_NAME	0x102B
CL_DEVICE_VENDOR	0x102C
CL_DRIVER_VERSION	0x102D
CL_DEVICE_PROFILE	0x102E
CL_DEVICE_VERSION	0x102F
CL_DEVICE_EXTENSIONS	0x1030
CL_DEVICE_PLATFORM	0x1031
CL_DEVICE_DOUBLE_FP_CONFIG	0x1032
CL_DEVICE_PREFERRED_VECTOR_WIDTH_HALF	0x1034
CL_DEVICE_HOST_UNIFIED_MEMORY	0x1035
CL_DEVICE_NATIVE_VECTOR_WIDTH_CHAR	0x1036
CL_DEVICE_NATIVE_VECTOR_WIDTH_SHORT	0x1037
CL_DEVICE_NATIVE_VECTOR_WIDTH_INT	0x1038
CL_DEVICE_NATIVE_VECTOR_WIDTH_LONG	0x1039
CL_DEVICE_NATIVE_VECTOR_WIDTH_FLOAT	0x103A
CL_DEVICE_NATIVE_VECTOR_WIDTH_DOUBLE	0x103B
CL_DEVICE_NATIVE_VECTOR_WIDTH_HALF	0x103C
CL_DEVICE_OPENCL_C_VERSION	0x103D
CL_DEVICE_LINKER_AVAILABLE	0x103E
CL_DEVICE_BUILT_IN_KERNELS	0x103F
CL_DEVICE_IMAGE_MAX_BUFFER_SIZE	0x1040
CL_DEVICE_IMAGE_MAX_ARRAY_SIZE	0x1041
CL_DEVICE_PARENT_DEVICE	0x1042
CL_DEVICE_PARTITION_MAX_SUB_DEVICES	0x1043
CL_DEVICE_PARTITION_PROPERTIES	0x1044
CL_DEVICE_PARTITION_AFFINITY_DOMAIN	0x1045

© 2000-2015, MetaQuotes Software Corp.

CL_DEVICE_PARTITION_TYPE	0x1046
CL_DEVICE_REFERENCE_COUNT	0x1047
CL_DEVICE_PREFERRED_INTEROP_USER_SYNC	0x1048
CL_DEVICE_PRINTF_BUFFER_SIZE	0x1049
CL_DEVICE_IMAGE_PITCH_ALIGNMENT	0x104A
CL_DEVICE_IMAGE_BASE_ADDRESS_ALIGNMENT	0x104B

Exemple:

```
void OnStart()
  {
//---
   int dCount= CLGetInfoInteger(0,CL_DEVICE_COUNT);
   for(int i = 0; i<dCount; i++)
     {
      int clCtx=CLContextCreate(i);
      if(clCtx == -1)
         Print("ERROR in CLContextCreate");
      string device;
      CLGetInfoString(clCtx,CL_DEVICE_NAME,device);
      Print(i,": ",device);
      uchar data[1024];
      uint size;
      CLGetDeviceInfo(clCtx,CL_DEVICE_VENDOR,data,size);
      Print("size = ",size);
      string str=CharArrayToString(data);
      Print(str);
     }
  }
//--- l'exemple de la sortie au journal "Experts"
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      2: Advanced Micro Devices, Inc.
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 32
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Tahiti
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Intel(R) Corporation
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 21
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      1:        Intel(R) Core(TM) i7-377
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      NVIDIA Corporation
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 19
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      0: GeForce GTX 580
```

© 2000-2015, MetaQuotes Software Corp.

CLProgramCreate

Crée le programme OpenCL du code initial.

```
int  CLProgramCreate(
   int           context,     // le handle sur le cotexte OpenCL
   const string  source       // le code initial
   );
```

Paramètres

context

[in] Le handle du cotexte OpenCL.

source

[in] La chaîne avec le code initial du programme OpenCL.

La valeur rendue

Le handle sur le tampon OpenCL à l'exécution réussie. Dans le cas de l'erreur rend -1. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

Pour ce moment on prévoit les codes suivants des erreurs:

- ERR_OPENCL_INVALID_HANDLE - le handle non valide sur le contexte OpenCL,
- ERR_INVALID_PARAMETER - le paramètre de chaîne non valide,
- ERR_NOT_ENOUGH_MEMORY - pas assez de mémoire pour terminer l'opération,
- ERR_OPENCL_PROGRAM_CREATE - l'erreur intérieure OpenCL ou l'erreur de la compilation.

In some graphic cards working with the double type numbers is disabled by default. This can lead to compilation error 5105. To enable support for the double type numbers, please add the following directive to your OpenCL program: #pragma OPENCL EXTENSION cl_khr_fp64 : enable

Example:

```
//+------------------------------------------------------------------+
//| OpenCL kernel                                                    |
//+------------------------------------------------------------------+
const string
cl_src=
       //--- by default some GPU doesn't support doubles
       //--- cl_khr_fp64 directive is used to enable work with doubles
       "#pragma OPENCL EXTENSION cl_khr_fp64 : enable      \r\n"
       //--- OpenCL kernel function
       "__kernel void Test_GPU(__global double *data,      \r\n"
       "                       const    int N,             \r\n"
       "                       const    int total_arrays)  \r\n"
       "  {                                                 \r\n"
       "   uint kernel_index=get_global_id(0);              \r\n"
       "   if (kernel_index>total_arrays) return;           \r\n"
       "   uint local_start_offset=kernel_index*N;          \r\n"
```

© 2000-2015, MetaQuotes Software Corp.

```
       "   for(int i=0; i<N; i++)                              \r\n"
       "     {                                                 \r\n"
       "       data[i+local_start_offset] *= 2.0;              \r\n"
       "     }                                                 \r\n"
       "  }                                                    \r\n";
//+------------------------------------------------------------------+
//| Test_CPU                                                         |
//+------------------------------------------------------------------+
bool Test_CPU(double &data[],const int N,const int id,const int total_arrays)
  {
//--- check array size
   if(ArraySize(data)==0) return(false);
//--- check array index
   if(id>total_arrays) return(false);
//--- calculate local offset for array with index id
   int local_start_offset=id*N;
//--- multiply elements by 2
   for(int i=0; i<N; i++)
     {
      data[i+local_start_offset]*=2.0;
     }
   return true;
  }
//---
#define ARRAY_SIZE   100  // size of the array
#define TOTAL_ARRAYS 5    // total arrays
//--- OpenCL handles
int cl_ctx;  // OpenCL context handle
int cl_prg;  // OpenCL program handle
int cl_krn;  // OpenCL kernel handle
int cl_mem;  // OpenCL buffer handle
//---
double DataArray1[]; // data array for CPU calculation
double DataArray2[]; // data array for GPU calculation
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart()
  {
//--- initialize OpenCL objects
//--- create OpenCL context
   if((cl_ctx=CLContextCreate())==INVALID_HANDLE)
     {
      Print("OpenCL not found. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL program
   if((cl_prg=CLProgramCreate(cl_ctx,cl_src))==INVALID_HANDLE)
     {
```

© 2000-2015, MetaQuotes Software Corp.

```
      CLContextFree(cl_ctx);
      Print("OpenCL program create failed. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL kernel
   if((cl_krn=CLKernelCreate(cl_prg,"Test_GPU"))==INVALID_HANDLE)
     {
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL kernel create failed. Error=",GetLastError());
      return(1);
     }
//--- create OpenCL buffer
   if((cl_mem=CLBufferCreate(cl_ctx,ARRAY_SIZE*TOTAL_ARRAYS*sizeof(double),CL_MEM_REAI
     {
      CLKernelFree(cl_krn);
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL buffer create failed. Error=",GetLastError());
      return(1);
     }
//--- set OpenCL kernel constant parameters
   CLSetKernelArgMem(cl_krn,0,cl_mem);
   CLSetKernelArg(cl_krn,1,ARRAY_SIZE);
   CLSetKernelArg(cl_krn,2,TOTAL_ARRAYS);
//--- prepare data arrays
   ArrayResize(DataArray1,ARRAY_SIZE*TOTAL_ARRAYS);
   ArrayResize(DataArray2,ARRAY_SIZE*TOTAL_ARRAYS);
//--- fill arrays with data
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculate local start offset for jth array
      uint local_offset=j*ARRAY_SIZE;
      //--- prepare array with index j
      for(int i=0; i<ARRAY_SIZE; i++)
        {
         //--- fill arrays with function MathCos(i+j);
         DataArray1[i+local_offset]=MathCos(i+j);
         DataArray2[i+local_offset]=MathCos(i+j);
        }
     };
//--- test CPU calculation
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculation of the array with index j
      Test_CPU(DataArray1,ARRAY_SIZE,j,TOTAL_ARRAYS);
     }
//--- prepare CLExecute params
   uint  offset[]={0};
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- global work size
   uint  work[]={TOTAL_ARRAYS};
//--- write data to OpenCL buffer
   CLBufferWrite(cl_mem,DataArray2);
//--- execute OpenCL kernel
   CLExecute(cl_krn,1,offset,work);
//--- read data from OpenCL buffer
   CLBufferRead(cl_mem,DataArray2);
//--- total error
   double total_error=0;
//--- compare results and calculate error
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- calculate local offset for jth array
      uint local_offset=j*ARRAY_SIZE;
      //--- compare the results
      for(int i=0; i<ARRAY_SIZE; i++)
        {
         double v1=DataArray1[i+local_offset];
         double v2=DataArray2[i+local_offset];
         double delta=MathAbs(v2-v1);
         total_error+=delta;
         //--- show first and last arrays
         if((j==0) || (j==TOTAL_ARRAYS-1))
            PrintFormat("array %d of %d, element [%d]:  %f, %f, [error]=%f",j+1,TOTAL_
        }
     }
   PrintFormat("Total error: %f",total_error);
//--- delete OpenCL objects
//--- free OpenCL buffer
   CLBufferFree(cl_mem);
//--- free OpenCL kernel
   CLKernelFree(cl_krn);
//--- free OpenCL program
   CLProgramFree(cl_prg);
//--- free OpenCL context
   CLContextFree(cl_ctx);
//---
   return(0);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CLProgramFree

Supprime le programme OpenCL.

```
void  CLProgramFree(
   int  program     // le handle sur l'objet OpenCL
   );
```

Paramètres

program

[in] Le handle de l'objet OpenCL.

La valeur rendue

Non. En cas de l'erreur intérieure change la valeur _LastError. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

CLKernelCreate

Crée le point de l'entrée au programme OpenCL et rend le handle sur lui.

```
int  CLKernelCreate(
   int           program,        // le handle sur l'objet OpenCL
   const string  kernel_name     // le nom de la fonction est "kernel"
   );
```

Paramètres

program

[in] Le handle sur l'objet du programme OpenCL.

kernel_name

[in] Le nom de la fonction est "kernel", c'est-à-dire le nom du point de l'entrée au programme OpenCL correspondant, par qui commence l'exécution.

La valeur rendue

Le handle sur le tampon OpenCL à l'exécution réussie. Dans le cas de l'erreur rend -1. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

Pour ce moment on prévoit les codes suivants des erreurs:

- ERR_OPENCL_INVALID_HANDLE - le handle non valide sur *program* OpenCL,
- ERR_INVALID_PARAMETER - le paramètre de chaîne non valide,
- ERR_OPENCL_TOO_LONG_KERNEL_NAME - le nom du kernel, contient plus de 127 caractères,
- ERR_OPENCL_KERNEL_CREATE - l'erreur intérieure à la création de l'objet OpenCL.

© 2000-2015, MetaQuotes Software Corp.

CLKernelFree

Supprime la fonction du lancement d' OpenCL.

```
void  CLKernelFree(
   int  kernel      // le handle sur le kernel du programme OpenCL
   );
```

Paramètres

kernel_name

[in] Le handle de l'objet du kernel.

La valeur rendue

Non. En cas de l'erreur intérieure change la valeur _LastError. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

CLSetKernelArg

Expose le paramètre de la fonction OpenCL.

```
bool  CLSetKernelArg(
   int   kernel,       // le handle sur le kernel du programme OpenCL
   uint  arg_index,    // le numéro de l'argument de la fonction OpenCL
   void  arg_value     // le code initial
   );
```

Paramètres

kernel

[in] Le handle sur le kernel du programme OpenCL.

arg_index

[in] Le numéro de l'argument de la fonction, le numérotage commence par zéro.

arg_value

[in] La valeur de l'argument de la fonction.

La valeur rendue

En cas de l'exécution successive rend true, autrement rend false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

Pour ce moment on prévoit les codes suivants des erreurs:

- ERR_INVALID_PARAMETER,
- ERR_OPENCL_INVALID_HANDLE - le handle non valide sur le kernel OpenCL,
- ERR_OPENCL_SET_KERNEL_PARAMETER - l'erreur intérieure OpenCL.

© 2000-2015, MetaQuotes Software Corp.

CLSetKernelArgMem

Expose le tampon OpenCL comme un paramètre de la fonction OpenCL.

```
bool  CLSetKernelArgMem(
   int   kernel,            // le handle sur le kernel du programme OpenCL
   uint  arg_index,         // le numéro de l'argument de la fonction OpenCL
   int   cl_mem_handle,     // le handle sur le tampon OpenCL
   );
```

Paramètres

kernel

[in] Le handle sur le kernel du programme OpenCL.

arg_index

[in] Le numéro de l'argument de la fonction, le numérotage commence par zéro.

cl_mem_handle

[in] Le handle sur le tampon OpenCL.

La valeur rendue

En cas de l'exécution successive rend true, autrement rend false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

CLBufferCreate

Crée le tampon OpenCL et rend le handle sur lui.

```
int  CLBufferCreate(
   int   context,     // le handle sur le contexte OpenCL
   uint  size         // la taille du tampon
   uint  flags        // les propriétés du tampon spécifiées par la combinaison des d
   );
```

Paramètres

context

[in] Le handle sur le contexte OpenCL.

size

[in] La taille du tampon en bytes.

flags

[in] Les propriétés du tampon spécifiées par la combinaison des drapeaux: CL_MEM_READ_WRITE, CL_MEM_WRITE_ONLY, CL_MEM_READ_ONLY, CL_MEM_ALLOC_HOST_PTR.

La valeur rendue

Le handle sur le tampon OpenCL à l'exécution réussie. Dans le cas de l'erreur rend -1. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

Pour ce moment on prévoit les codes suivants des erreurs:

- ERR_OPENCL_INVALID_HANDLE - le handle non valide sur le contexte OpenCL,
- ERR_NOT_ENOUGH_MEMORY - pas assez de mémoire,
- ERR_OPENCL_BUFFER_CREATE - l'erreur intérieure de la création du tampon.

© 2000-2015, MetaQuotes Software Corp.

CLBufferFree

Supprime le tampon OpenCL.

```
void  CLBufferFree(
   int   buffer     // le handle sur le tampon OpenCL
   );
```

Paramètres

buffer

[in] Le handle sur le tampon OpenCL.

La valeur rendue

Non. En cas de l'erreur intérieure change la valeur _LastError. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

CLBufferWrite

Inscrit le tableau au tampon OpenCL et rend la quantité d'éléments inscrits.

```
uint  CLBufferWrite(
   int           buffer,                  // le handle sur le tampon OpenCL
   const void&   data[],                  // le tableau des valeurs
   uint          buffer_offset=0,         // le décalage dans le tampon OpenCL en oct
   uint          data_offset=0,           // le décalage dans le tampon dans les élér
   uint          data_count=WHOLE_ARRAY   // le nombre des valeurs du tableau pour l'
   );
```

Paramètres

buffer

[in] Le handle du tampon OpenCL.

data[]

[in] le tableau des valeurs, lesquelles il est nécessaire d'inscrire au tampon OpenCL. est transmis selon la référence.

buffer_offset

[in] le décalage dans le tampon OpenCL en octets, par lequel commence l'inscription. Par défaut l'inscription va dès le début du tampon.

data_offset

[in] L'index du premier élément du tableau, à partir de lequel les valeurs sont tirées du tableau pour l'inscription au tampon OpenCL. Par défaut on prend les valeurs dès le début du tableau.

data_count

[in] Le nombre de valeurs, lesquelles il faut inscrire. Toutes les valeurs du tableau par défaut.

La valeur rendue

Le nombre d'éléments inscrits, en cas de l'erreur 0 se revient. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

Pour les tableaux unidimentionnels le numéro de l'élément, par qui commence la lecture des données pour l'inscription au tampon OpenCL, est calculé en prenant en compte le drapeau AS_SERIES.

Le tableau avec la dimension deux et plus est présenté comme unidimentionnel. Dans ce cas *data_offset* - c'est une quantité d'éléments, qu'il faut manquer dans la représentation, et non le nombre d'éléments dans une première dimension.

© 2000-2015, MetaQuotes Software Corp.

CLBufferRead

Lit le tampon OpenCL au tableau et rend la quantité d'éléments lus.

```
uint  CLBufferRead(
   int           buffer,                  // le handle sur le tampon OpenCL
   const void&   data[],                  // le tableau des valeurs
   uint          buffer_offset=0,         // le décalage dans le tampon OpenCL en oct
   uint          data_offset=0,           // le décalage dans le tampon dans les élér
   uint          data_count=WHOLE_ARRAY   // le nombre des valeurs du tableau pour la
   );
```

Paramètres

buffer

[in] Le handle du tampon OpenCL.

data[]

[in] Le massif pour la réception des valeurs du tampon OpenCL. est transmis selon la référence.

buffer_offset

[in] Le décalage dans le tampon OpenCL en octets, par lequel commence la lecture. Par défaut la lecture commence par le début du tampon.

data_offset

[in] L'index du premier élément du massif pour l'inscription des valeurs du tampon OpenCL. Par défaut la lecture l'inscription des valeurs lues au tableau commence par index nul.

data_count

[in] Le nombre de valeurs, lesquelles il faut lire. Tout le tampon OpenCL est lu par défaut.

La valeur rendue

Le nombre d'éléments lus, en cas de l'erreur 0 se revient. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

Note

Pour les tableaux unidimentionnels le numéro de l'élément, auquel commence l'inscription des données du tampon OpenCL, est calculé en prenant en compte le drapeau AS_SERIES.

Le tableau avec la dimension deux et plus est présenté comme unidimentionnel. Dans ce cas *data_offset* - c'est une quantité d'éléments, qu'il faut manquer dans la représentation, et non le nombre d'éléments dans une première dimension.

© 2000-2015, MetaQuotes Software Corp.

CLExecute

Exécute le programme OpenCL. Il y a 3 variantes de la fonction:

1. Le lancement de la fonction kernel sur un seul noyau

```
bool  CLExecute(
   int          kernel                  // le handle sur le kernel du programme OpenC
   );
```

2. Le lancement de quelques copies kernel (la fonction OpenCL) avec une description de l'espace des tâches

```
bool  CLExecute(
   int          kernel,                 // le handle sur le kernel du programme Oper
   uint         work_dim,               // la dimension de l'espace des tâches
   const uint&  global_work_offset[],   // le décalage initial dans l'espace des tâc
   const uint&  global_work_size[]      // le nombre total de tâches
   );
```

3. Le lancement de quelques copies kernel (la fonction OpenCL) avec la description de l'espace des tâches et l'indication de la taille de la sous-multitude locale des tâches dans le groupe

```
bool  CLExecute(
   int          kernel,                 // le handle sur le kernel du programme Oper
   uint         work_dim,               // la dimension de l'espace des tâches
   const uint&  global_work_offset[],   // le décalage initial dans l'espace des tâc
   const uint&  global_work_size[],     // le nombre total de tâches
   const uint&  local_work_size[]       // le nombre de tâches dans le groupe local
   );
```

Paramètres

kernel

[in] Le handle sur le kernel OpenCL.

work_dim

[in] La dimension de l'espace des tâches.

global_work_offset[]

[in] Le décalage initial dans l'espace des tâches.

global_work_size[]

[in] La taille d'un sous-ensemble de tâches.

local_work_size[]

[in] La taille de la sous-multitude locale des tâches dans le groupe.

La valeur rendue

En cas de l'exécution successive rend true, autrement rend false. Pour recevoir l'information sur l'erreur, utilisez la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

Note

montrons le sens des paramètres à l'exemple:

- *work_dim* spécifie la dimension du tableau *work_items[]*, décrivant les tâches. Si *work_dim*=3, on utilise le tableau en 3 dimensions *work_items[N1, N2, N3]*.
- *global_work_size[]* contient des valeurs qui spécifient la taille du tableau *work_items[]*. Si work_dim=3, et en conséquence le tableau*global_work_size[3]* peut-être {40, 100, 320}. Alors nous avons *work_items[40, 100, 320]*. Ainsi, le nombre total de tâches est égal à 40 x 100 x 320 = 1 280 000.
- *local_work_size[]* spécifie un sous-ensemble des tâches à exécuter par le noyau OpenCL spécifié du programme. Sa dimension est égale à la dimension *work_items[]* et permet de couper la sous-multitude totale des tâches sur les sous-multitudes plus petites sans restes de la division. En effet, les dimensions du tableau *local_work_size[]* doivent être choisis de sorte que la pluralité de tâches globales *work_items[]* soient coupées en petits sous-ensembles. Dans l'exemple donné convient *local_work_size[3]={10, 10, 10}*, car *work_items[40, 100, 320]*on peut ramasser du tableau*local_items[10, 10, 10]*sans reste.

© 2000-2015, MetaQuotes Software Corp.

Bibliothèque Standard

Cet ensemble de chapitres contient les détails techniques de la Bibliothèque Standard MQL5 et la descriptions de ses composants principaux.

La Bibliothèque Standard MQL5 est écrite en MQL5 et est conçue pour faciliter l'écriture des programmes (indicateurs, scripts, experts) aux utilisateurs. La Bibliothèque fournit un accès aisé à la plupart des fonctions internes MQL5.

La Bibliothèque Standard MQL5 est située dans le répertoire de travail du terminal dans le répertoire Include.

Section	Localisation
Classe de base	Include\
Stub" pour la méthode de changement de code	Include\Arrays\
Classes pour les opérations sur les fichiers	Include\Files\
Classes pour les opérations sur les chaines de caractères	Include\Strings\
Classes pour les objets graphiques	Include\Objects\
Classe pour créer des objets graphiques	Include\Canvas\
Classe pour travailler avec un graphique	Include\Charts\
Indicateurs techniques	Include\Indicators\
Classes de trading	Include\Trade\
Classes des stratégies de trading	Include\Expert\
Classes de création des panneaux de contrôle et des dialogues	Include\Controls\

© 2000-2015, MetaQuotes Software Corp.

Classe de Base CObject

La classe CObject est la classe de base de la Bibliothèque Standard MQL5.

Description

La classe CObject permet à tous ses descendants de faire partie d'une liste chaînée. Elle propose également différentes méthodes virtuelles à implémenter dans ses classes descendantes.

Déclaration

```
class CObject
```

Titre

```
#include <Object.mqh>
```

Méthodes de Classe

Attributs	
Prev	Retourne la valeur de l'élèment précédent
Prev	Définit la valeur de l'élément précédent
Next	Retourne la valeur de l'élèment suivant
Next	Définit la valeur de l'élément suivant
Méthodes de comparaison	
virtual Compare	Retourne le résultat de la comparaison avec un autre objet
Entrée/Sortie	
virtual Save	Ecrit l'objet dans un fichierf
virtual Load	Charge l'objet depuis un fichier
virtual Type	Retourne le type de l'objet

Classes dérivées :

- CArray
- CChartObject
- CChart
- CString
- CFile
- CList
- CTreeNode

© 2000-2015, MetaQuotes Software Corp.

Prev

Retourne un pointeur sur l'élément précédent de la liste.

```
CObject*  Prev()
```

Valeur de Retour

Pointeur sur l'élément précédent de la liste. Si l'élément courant est le premier élément de la liste, retourne NULL.

Exemple :

```
//--- exemple d'utilisation de CObject::Prev()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- définit le lien entre les éléments
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utilise l'objet précédent
   CObject *object=object_second.Prev();
   //--- supprime les objets
   delete object_first;
   delete object_second;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Prev

Définit le pointeur vers l'élément précédent de la liste.

```
void  Prev(
   CObject*  object      // Pointeur vers l'élément précédent de la liste
   )
```

Paramètres

object

[in] Nouvelle valeur du pointeur sur l'élément précédent de la liste.

Exemple :

```
//--- exemple d'utilisation de CObject::Prev(CObject*)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- définit le lien entre les éléments
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utilise les objets
   //--- ...
   //--- supprime les objets
   delete object_first;
   delete object_second;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Next

Retourne un pointeur sur l'élément suivant de la liste.

```
CObject*  Next()
```

Valeur de Retour

Pointeur sur l'élément suivant de la liste. Si l'élément courant est le dernier élément de la liste, retourne NULL.

Exemple :

```
//--- exemple d'utilisation de CObject::Next()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- définit le lien entre les éléments
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utilise l'objet suivant de la liste
   CObject *object=object_first.Next();
   //--- supprime les objets
   delete object_first;
   delete object_second;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Next

Définit le pointeur sur l'élément suivant de la liste.

```
void  Next(
   CObject*  object      // Pointeur sur l'élément suivant de la liste
   )
```

Paramètres

object

[in] Nouvelle valeur du pointeur sur l'élément suivant de la liste.

Exemple :

```
//--- exemple d'utilisation de CObject::Next(CObject*)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- définit le lien entre les éléments
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- utilise les objets
   //--- ...
   //--- supprime les objets
   delete object_first;
   delete object_second;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Compare

Compare les données d'un élément de la liste avec les données d'un autre élément de la liste.

```
virtual int  Compare(
   CObject const *  node,     // Node avec lequel comparer l'objet
   int              mode=0    // Mode de comparaison
   ) const
```

Paramètres

node

[in] Pointeur vers un élément de la liste à comparer

mode=0

[in] Mode de comparaison

Valeur de Retour

0 - si les 2 éléments de la liste sont identiques, -1 - si l'élément de la liste est inférieur à l'élément de comparaison (node), 1 - si l'élément de la liste est supérieur à l'élément de comparaison (node).

Note

La méthode Compare () de la classe CObject retourne toujours 0 et n'exécute aucune action. Si vous désirez comparer des données de classes dérivées, la méthode Compare (...) doit être implémentée. Le paramètre 'mode' devrait être utilisé dans le cas de comparaison où plusieurs variantes sont possibles.

Exemple :

```
//--- exemple d'utilisation de CObject::Compare(...)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- définit le lien entre les éléments
```

© 2000-2015, MetaQuotes Software Corp.

```
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- compare les objets
   int result=object_first.Compare(object_second);
   //--- supprime les objets
   delete object_first;
   delete object_second;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde un élément et ses données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

La méthode Save (int) de la classe CObject retourne toujours vrai et n'exécute aucune action. Si vous désirez sauvegarder des données de classes dérivées depuis un fichier, la méthode Save (int) doit être implémentée.

Exemple :

```
//--- exemple d'utilisation de CObject::Save(int)
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- définit les données de l'objet
   //--- . . .
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
```

© 2000-2015, MetaQuotes Software Corp.

```
         }
       FileClose(file_handle);
      }
   delete object;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge un élément et ses données dans la liste depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

La méthode Load (int) de la classe CObject retourne toujours vrai et n'exécute aucune action. Si vous désirez charger des données de classes dérivées depuis un fichier, la méthode Load (int) doit être implémentée.

Exemple :

```
//--- exemple d'utilisation de CObject::Load(int)
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2015, MetaQuotes Software Corp.

```
    }
  //--- utilise l'objet
  //--- . . .
  delete object;
}
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type de l'objet (pour CObject - 0).

Exemple :

```
//--- exemple d'utilisation de CObject::Type()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object=new CObject;
   //---
   object=new CObject;
   if(object ==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type de l'objet
   int type=object.Type();
   //--- supprime l'objet
   delete object;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Structures de Données

Cette section contient les détails techniques d'utilisation des différentes structures de données (tableaux, listes chaînées, etc.) et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Utiliser des classes de structures de données vous fera gagner du temps lors de la création et du stockage de vos données dans différents formats (incluant les structures de données composites).

La Bibliothèque Standard MQL5 (en terme de données) est située dans le répertoire de travail du terminal dans le répertoire Include\Arrays.

Tableaux de Données

L'utiisation de classes de tableaux de données dynamiques vous fera gagner du temps lors de la création et du stockage de différents formats (incluant les tableaux à plusieurs dimensions).

La Bibliothèque Standard MQL5 (en terme de tableaux de données) est située dans le répertoire de travail du terminal dans le répertoire Include\Arrays.

Classe	Description
Classe de base du tableau dynamique de données CArray	Classe de base du tableau dynamique de données
CArrayChar	Tableau dynamique de variables de type char ou uchar
CArrayShort	Tableau dynamique de variables de type short ou ushort
CArrayInt	Tableau dynamique de variables de type int ou uint
CArrayLong	Tableau dynamique de variables de type long ou ulong
CArrayFloat	Tableau dynamique de variables de type float
CArrayDouble	Tableau dynamique de variables de type double
CArrayString	Tableau dynamique de variables de type string
Classe de base du tableau d'objet CArrayObj	Tableau dynamique de pointeurs de CObject
Classe de base de la liste CList	Fournit la possibilité de travailler avec une liste d'instance de CObject et de ses descendants
CTreeNode	Fournit la possibilité de travailler avec les noeuds de l'arbre binaire CTree
CTree	Fournit la possibilité de travailler avec l'arbre binaire d'instances de classe de CTreeNode et de ses descendants

© 2000-2015, MetaQuotes Software Corp.

CArray

La classe CArray est la classe de base des tableaux dynamiques de variables.

Description

La classe CArray permet d'effectuer des opérations sur les tableaux dynamiques de variables : allocation mémoire, tri et chargement/sauvegarde.

Déclaration

```
class CArray : public CObject
```

Titre

```
#include <Arrays\Array.mqh>
```

Méthodes de Classe

Attributs	
Step	Retourne la taille du pas d'incrément du tableau
Step	Définit la taille du pas d'incrément du tableau
Total	Retourne le nombre total d'éléments dans le tableau
Available	Retourne le nombre d'éléments libres dans le tableau sans besoin d'allocation mémoire supplémentaire
Max	Retourne la taille possible maximum du tableau sans réallocation de mémoire
IsSorted	Retourne le tableau trié selon l'option spécifiée
SortMode	Retourne le mode de tri du tableau
Méthodes de suppression	
Clear	Supprime tous les éléments du tableau sans désallocation de la mémoire
Méthodes de tri	
Sort	Trie un tableau selon l'option spécifiée
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.

© 2000-2015, MetaQuotes Software Corp.

Classes dérivées :

- CArrayChar
- CArrayShort
- CArrayInt
- CArrayLong
- CArrayFloat
- CArrayDouble
- CArrayString
- CArrayObj

© 2000-2015, MetaQuotes Software Corp.

Step

Retourne la taille du pas d'incrément du tableau.

```
int  Step() const
```

Valeur de Retour

Taille d'incrément du tableau.

Exemple :

```
//--- exemple d'utilisation de CArray::Step()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le pas
   int step=array.Step();
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Step

Définit la taille du pas d'incrément du tableau.

```
bool  Step(
   int  step      // pas
   )
```

Paramètres

step

[in] La nouvelle valeur du pas d'incrément du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le pas est inférieur ou égal à zéro.

Exemple :

```
//--- exemple d'utilisation de CArray::Step(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- définit la taille du pas
   bool result=array.Step(1024);
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Total

Retourne le nombre total d'éléments dans le tableau.

```
int  Total() const;
```

Valeur de Retour

Nombre d'éléments du tableau.

Exemple :

```
//--- exemple d'utilisation de CArray::Total()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie le total
   int total=array.Total();
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Available

Retourne le nombre d'éléments libres dans le tableau sans besoin d'allocation mémoire supplémentaire

```
int  Available() const
```

Valeur de Retour

Nombre d'éléments libres dans le tableau sans besoin d'allocation mémoire supplémentaire

Exemple :

```
//--- exemple d'utilisation de CArray::Available()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie les éléments disponibles
   int available=array.Available();
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Max

Retourne la taille possible maximum du tableau sans réallocation de mémoire

```
int  Max() const
```

Valeur de Retour

La taille possible maximum du tableau sans réallocation de la mémoire.

Exemple :

```
//--- exemple d'utilisation de CArray::Max()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie la taille maximum
   int max=array.Max();
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

IsSorted

Retourne le tableau trié selon l'option spécifiée.

```
bool  IsSorted(
   int  mode=0      // Mode de tri
   ) const
```

Paramètres

mode=0

[in] Mode de tri.

Valeur de Retour

Flag de la liste triée. Vrai si le tableau est trié suivant le mode spécifié, faux sinon.

Note

Le mode de tri du tableau ne peut pas être changé directement. La méthode Sort() réinitialise toutes les méthodes pour ajouter/insérer, excepté InsertSort(...).

Exemple :

```
//--- exemple d'utilisation de CArray::IsSorted()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie si le tableau est trié
   if(array.IsSorted())
     {
      //--- utilisation de méthodes sur le tableau trié
      //--- ...
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SortMode

Retourne le mode de tri du tableau.

```
int  SortMode() const;
```

Valeur de Retour

Mode de tri.

Exemple :

```
//--- exemple d'utilisation de CArray::SortMode()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie le mode du tri
   int sort_mode=array.SortMode();
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Clear

Supprime tous les éléments du tableau sans désallocation de la mémoire

```
void  Clear()
```

Valeur de Retour

Aucune.

Exemple :

```
//--- exemple d'utilisation de CArray::Clear()
#include <Arrays\Array.mqh>
//---
void OnStart()
 {
 CArray *array=new CArray;
 //---
 if(array==NULL)
 {
 printf("Object create error");
 return;
 }
 //--- utilisation du tableau
 //--- ...
 //--- nettoyage du tableau
 array.Clear();
 //--- supprime le tableau
 delete array;
 }
```

© 2000-2015, MetaQuotes Software Corp.

Sort

Trie un tableau selon l'option spécifiée.

```
void  Sort(
   int  mode=0      // Mode de tri
   )
```

Paramètres

mode=0

[in] Mode de tri du tableau.

Valeur de Retour

Aucune.

Note

Le tri d'un tableau est toujours fait en mode ascendant (du plus petit au plus grand). Pour les tableaux de types primitifs (CArrayChar, CArrayShort, etc.), le mode paramètre n'est pas utilisé. Pour le tableau CArrayObj, les différentes méthodes de tri devraient être implémentées dans la fonction Sort (int) des classes dérivées.

Exemple :

```
//--- exemple d'utilisation de CArray::Sort(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- tri par mode 0
   array.Sort(0);
   //--- utilise le tableau
   //--- ...
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArray::Save(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArray::Load(...)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CArrayChar

La classe CArrayChar est la classe des tableaux dynamiques de variables de type char ou uchar.

Description

La classe CArrayChar permet de travailler avec les tableaux dynamiques de variables de type char ou uchar. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayChar : public CArray
```

Titre

```
#include <Arrays\ArrayChar.mqh>
```

Méthodes de Classe

Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau
Méthodes de modification	

© 2000-2015, MetaQuotes Software Corp.

Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Méthodes concernant les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Cherche un élément égal à celui donné dans un tableau trié
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.

© 2000-2015, MetaQuotes Software Corp.

virtual Type	Retourne l'identifiant du type du tableau

© 2000-2015, MetaQuotes Software Corp.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Reserve(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si la taille passée en argument est inférieure ou égale à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Resize(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Shutdown()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   char  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Add(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const char&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::AddArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayChar*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayChar - source des éléments à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::AddArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   char  element,     // Elément à insérer
   int   pos          // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Insert(char,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const char&  src[],     // Tableau à insérerS
   int           pos       // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::InsertArray(const char &[],int)
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayChar*  src,      // Pointeur vers le tableau à insérer
   int           pos      // Position
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayChar - source des éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::InsertArray(const CArrayChar*,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2015, MetaQuotes Software Corp.

```
      return;
     }
   //--- supprime le tableau source
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const char&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::AssignArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayChar*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayChar - source des éléments à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::AssignArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayChar *src  =new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2015, MetaQuotes Software Corp.

```
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int   pos,          // Position
   char  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Update(int,char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,'A'))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer dans le tableau

shift

[in] Valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Shift(int,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Delete(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,     // Position du permier élément
   int  to        // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément à supprimer dans le tableau.

to

[in] Position du dernier élément à supprimer dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::DeleteRange(int,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
char  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, CHAR_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

CHAR_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::At(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      char result=array.At(i);
      if(result==CHAR_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const char&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::CompareArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const CArrayChar*  src      // Pointeur vers le tableau à comparer
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de la classe CArrayChar - source des éléments à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::CompareArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   char  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::InsertSort(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort('A'))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Search(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search('A')!=-1) printf("Element found");
   else                      printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchGreat(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat('A')!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchLess(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess('A')!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié

```
int  SearchGreatOrEqual(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchGreatOrEqual(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual('A')!=-1) printf("Element found");
   else                                  printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchLessOrEqual(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual('A')!=-1) printf("Element found");
   else                                 printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchFirst(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst('A')!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchLast(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast('A')!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   char  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::SearchLinear(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear('A')!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Save(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Load(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%c'",i,array.At(i));
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (77 pour CArrayChar).

Exemple :

```
//--- exemple d'utilisation de CArrayChar::Type()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CArrayShort

La classe CArrayLong est la classe des tableaux dynamiques de variables de type short ou ushort.

Description

La classe CArrayShort permet de travailler avec les tableaux dynamiques de variables de type short ou ushort. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayShort : public CArray
```

Titre

```
#include <Arrays\ArrayShort.mqh>
```

Méthodes de Classe

Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau
Méthodes de mise à jour	

© 2000-2015, MetaQuotes Software Corp.

Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Opérations avec les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.

© 2000-2015, MetaQuotes Software Corp.

virtual Type	Retourne l'identifiant du type du tableau

© 2000-2015, MetaQuotes Software Corp.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Reserve(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si la taille passée en argument est inférieure ou égale à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Resize(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Shutdown()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   short  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Add(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const short&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::AddArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayShort*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayShort contenant les éléments source à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::AddArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   short  element,     // Elément à insérer
   int    pos          // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Insert(short,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const short&  src[],     // Tableau source
   int            pos        // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::InsertArray(const short &[],int)
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayShort*  src,     // Pointeur vers la source
   int            pos      // Position
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayShort contenant les éléments source à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::InsertArray(const CArrayShort*,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2015, MetaQuotes Software Corp.

```
      return;
     }
   //--- supprime le tableau source
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const short&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::AssignArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayShort*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayShort contenant les éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::AssignArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayShort *src  =new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2015, MetaQuotes Software Corp.

```
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int    pos,          // Position
   short  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Update(int,short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,100))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Positions
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer dans le tableau

shift

[in] Valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Shift(int,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Delete(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,     // Position du permier élément
   int  to        // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément à supprimer dans le tableau.

to

[in] Position du dernier élément à supprimer dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::DeleteRange(int,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
short  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, SHORT_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

SHORT_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::At(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      short result=array.At(i);
      if(result==SHORT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const short&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::CompareArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const CArrayShort*  src      // Pointeur vers la source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de la classe CArrayShort contenant les éléments sources de comparaison.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::CompareArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   short  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::InsertSort(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(100))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Search(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search(100)!=-1) printf("Element found");
   else                      printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchGreat(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat(100)!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchLess(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess(100)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchGreatOrEqual(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(100)!=-1) printf("Element found");
   else                                  printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchLessOrEqual(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(100)!=-1) printf("Element found");
   else                                 printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchFirst(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst(100)!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchLast(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast(100)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   short  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::SearchLinear(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear(100)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Save(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Load(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (82 pour CArrayShort).

Exemple :

```
//--- exemple d'utilisation de CArrayShort::Type()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CArrayInt

La classe CArrayInt est la classe des tableaux dynamiques de variables de type int ou uint.

Description

La classe CArrayInt permet de travailler avec les tableaux dynamiques de variables de type int ou uint. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayInt : public CArray
```

Titre

```
#include <Arrays\ArrayInt.mqh>
```

Méthodes de Classe

Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau
Méthodes de mise à jour	

© 2000-2015, MetaQuotes Software Corp.

Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Opérations avec les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.

© 2000-2015, MetaQuotes Software Corp.

virtual Type	Retourne l'identifiant du type du tableau

© 2000-2015, MetaQuotes Software Corp.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Reserve(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si la taille passée en argument est inférieure ou égale à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Resize(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Shutdown()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   int  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Add(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const int&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::AddArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayInt*  src      // Pointeur vers le tableau source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayInt - source des éléments à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::AddArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   int  element,     // Elément à insérer
   int  pos          // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Insert(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const int&  src[],     // Tableau source
   int          pos        // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::InsertArray(const int &[],int)
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayInt*  src,     // Pointeur vers le tableau source
   int          pos      // Position
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayInt - source des éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::InsertArray(const CArrayInt*,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2015, MetaQuotes Software Corp.

```
      return;
     }
   //--- supprime le tableau source
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const int&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::AssignArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayInt*  src      // Pointeur vers le tableau source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayInt - source des éléments à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::AssignArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayInt *src  =new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- les tableaux sont identiques
   //--- supprime le tableau source
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int  pos,          // Position
   int  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Update(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,10000))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer dans le tableau

shift

[in] Valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Shift(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Delete(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,     // Position du permier élément
   int  to        // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément à supprimer dans le tableau.

to

[in] Position du dernier élément à supprimer dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::DeleteRange(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
int  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, INT_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

INT_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::At(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      int result=array.At(i);
      if(result==INT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const int&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::CompareArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const CArrayInt*  src      // Pointeur vers le tableau source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de la classe CArrayInt - source des éléments à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::CompareArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   int  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::InsertSort(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(10000))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Search(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search(10000)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchGreat(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat(10000)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchLess(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess(10000)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   int  element      // Elément à rechercher
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchGreatOrEqual(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(10000)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchLessOrEqual(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(10000)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt:: SearchFirst(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst(10000)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchLast(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast(10000)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   int  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::SearchLinear(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear(10000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Save(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Load(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (82 pour CArrayInt).

Exemple :

```
//--- exemple d'utilisation de CArrayInt::Type()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CArrayLong

La classe CArrayLong est la classe des tableaux dynamiques de variables de type long ou ulong.

Description

La classe CArrayLong permet de travailler avec les tableaux dynamiques de variables de type long ou ulong. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayLong : public CArray
```

Titre

```
#include <Arrays\ArrayLong.mqh>
```

Méthodes de Classe

Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau
Méthodes de mise à jour	

© 2000-2015, MetaQuotes Software Corp.

Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Opérations avec les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.

© 2000-2015, MetaQuotes Software Corp.

virtual Type	Retourne l'identifiant du type du tableau

© 2000-2015, MetaQuotes Software Corp.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Reserve(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si la taille passée en argument est inférieure ou égale à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Resize(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Shutdown()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   long  element      // Elément à insérer
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Add(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const long&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::AddArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayLong*  src      // Pointeur vers le tableau source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayLong - source des éléments à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::AddArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   long  element,     // Elément à insérer
   int   pos          // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Insert(long,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const long&  src[],     // Tableau source
   int           pos        // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::InsertArray(const long &[],int)
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayLong*  src,      // Pointeur vers la source
   int           pos      // Position
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayLong - source des éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::InsertArray(const CArrayLong*,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2015, MetaQuotes Software Corp.

```
      return;
     }
   //--- supprime le tableau source
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const long&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::AssignArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayLong*  src      // Pointeur vers le tableau source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayLong - source des éléments à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::AssignArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayLong *src  =new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2015, MetaQuotes Software Corp.

```
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int   pos,          // Position
   long  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Update(int,long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,1000000))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer dans le tableau

shift

[in] Valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Shift(int,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Delete(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,     // Position du permier élément
   int  to        // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément à supprimer dans le tableau.

to

[in] Position du dernier élément à supprimer dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::DeleteRange(int,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
long  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, LONG_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

LONG_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::At(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      long result=array.At(i);
      if(result==LONG_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  Erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const long&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::CompareArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArrayconst

Compare le tableau avec un autre tableau.

```
bool  CompareArrayconst(
   const CArrayLong*  src      // Pointeur vers le tableau source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de la classe CArrayLong - source des éléments à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::CompareArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   long  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::InsertSort(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(1000000))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Search

Recherche un élément du tableau égal au modèle donné dans un tableau trié.

```
int  Search(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Search(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search(1000000)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreat

Cherche un élément supérieur au modèle donné dans un tableau trié.

```
int  SearchGreat(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchGreat(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat(1000000)!=-1) printf("Element found");
   else                               printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchLess(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess(1000000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchGreatOrEqual(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(1000000)!=-1) printf("Element found");
   else                                      printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchLessOrEqual(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(1000000)!=-1) printf("Element found");
   else                                     printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchFirst

Cherche le premier élément égal au modèle dans un tableau trié.

```
int  SearchFirst(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchFirst(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst(1000000)!=-1) printf("Element found");
   else                               printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchLast(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast(1000000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLinear

Recherche un élément du tableau égal au modèle donné dans le tableau.

```
int  SearchLinear(
   long  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::SearchLinear(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear(1000000)!=-1) printf("Element found");
   else                                printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Save(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Load(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %I64",i,array.At(i));
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (84 pour CArrayLong).

Exemple :

```
//--- exemple d'utilisation de CArrayLong::Type()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CArrayFloat

La classe CArrayFloat est la classe des tableaux dynamiques de variables de type float.

Description

La classe CArrayFloat permet de travailler avec les tableaux dynamiques de variables de type float. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayFloat : public CArray
```

Titre

```
#include <Arrays\ArrayFloat.mqh>
```

Méthodes de Classe

Attributs	
Delta	Définit la tolérance de comparaison
Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau

© 2000-2015, MetaQuotes Software Corp.

AssignArray	Copie les éléments d'un autre tableau
Méthodes de mise à jour	
Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Opérations avec les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un

© 2000-2015, MetaQuotes Software Corp.

	fichier.
virtual Type	Retourne l'identifiant du type du tableau

© 2000-2015, MetaQuotes Software Corp.

Delta

Définit la tolérance de comparaison.

```
void  Delta(
   float  delta      // Tolérance
   )
```

Paramètres

delta

[in] La nouvelle valeur de tolérance de comparaison.

Valeur de Retour

Aucune

Note

Validité de la comparaison lors de la recherche. Les valeurs sont considérées égales si leur différence est inférieure ou égale à la tolérance. La tolérance par défaut est 0.0.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Delta(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- définit la tolérance de comparaison
   array.Delta(0.001);
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Reserve(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si la taille passée en argument est inférieure ou égale à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Resize(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Shutdown()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   float  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Add(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const float&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::AddArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayFloat*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayFloat - source des éléments à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::AddArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   float  element,     // Elément à insérer
   int    pos          // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Insert(float,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const float&  src[],     // Tableau source
   int            pos        // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::InsertArray(const float &[],int)
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayFloat*  src,     // Pointeur vers le tableau source
   int            pos      // Position
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayFloat - source des éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::InsertArray(const CArrayFloat*,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2015, MetaQuotes Software Corp.

```
      return;
     }
   //--- supprime le tableau source
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const float&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::AssignArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayFloat*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayFloat - source des éléments à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::AssignArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayFloat *src  =new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2015, MetaQuotes Software Corp.

```
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int    pos,          // Position
   float  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Update(int,float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,100.0))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer dans le tableau

shift

[in] Valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Shift(int,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Delete(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,     // Position du permier élément
   int  to        // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément à supprimer dans le tableau.

to

[in] Position du dernier élément à supprimer dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::DeleteRange(int,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
float  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, FLT_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

FLT_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::At(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      float result=array.At(i);
      if(result==FLT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  Erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const float&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::CompareArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArrayconst

Copie les éléments d'un autre tableau.

```
bool  AssignArrayconst(
   const CArrayFloat*  src      // Pointeur vers la source
   ) const
```

Paramètres

src

[in] Pointeur vers l'instance d'une classe CArrayFloat - source des éléments à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::CompareArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   float  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::InsertSort(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(100.0))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Search(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search(100.0)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchGreat(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat:: SearchLess(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchGreatOrEqual(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchLessOrEqual(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchFirst(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchLast(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   float  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::SearchLinear(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear(100.0)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Save(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Load(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (87 pour CArrayFloat).

Exemple :

```
//--- exemple d'utilisation de CArrayFloat::Type()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CArrayDouble

La classe CArrayDouble est la classe des tableaux dynamiques de variables de type double.

Description

La classe CArrayDouble permet de travailler avec les tableaux dynamiques de variables de type double. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayDouble : public CArray
```

Titre

```
#include <Arrays\ArrayDouble.mqh>
```

Méthodes de Classe

Attributs	
Delta	Définit la tolérance de comparaison
Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau

© 2000-2015, MetaQuotes Software Corp.

AssignArray	Copie les éléments d'un autre tableau
Méthodes de mise à jour	
Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Recherche du min/max	
Minimum	Retourne l'index du plus petit élément du tableau dans l'intervalle spécifié
Maximum	Retourne l'index du plus grand élément du tableau dans l'intervalle spécifié
Opérations avec les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié

© 2000-2015, MetaQuotes Software Corp.

SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.
virtual Type	Retourne l'identifiant du type du tableau

Classes dérivées :

- CIndicatorBuffer

© 2000-2015, MetaQuotes Software Corp.

Delta

Définit la tolérance de comparaison.

```
void  Delta(
   double  delta      // Tolérance
   )
```

Paramètres

delta

[in] La nouvelle valeur de tolérance de comparaison.

Valeur de Retour

Aucune

Note

Validité de la comparaison lors de la recherche. Les valeurs sont considérées égales si leur différence est inférieure ou égale à la tolérance. La tolérance par défaut est 0.0.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Delta(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- définit la tolérance de comparaison
   array.Delta(0.001);
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Reserve(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si la taille passée en argument est inférieure ou égale à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Resize(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Shutdown()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   double  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter au tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Add(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const double&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur le tableau des éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::AddArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayDouble*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayDouble contenant les éléments source à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::AddArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   double  element,     // Elément à insérer
   int     pos          // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Insert(double,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const double&  src[],     // Tableau source
   int            pos        // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::InsertArray(const double &[],int)
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayDouble*  src,     // Pointeur vers la source
   int            pos      // Position
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayDouble contenant les éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::InsertArray(const CArrayDouble*,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2015, MetaQuotes Software Corp.

```
      return;
     }
   //--- supprime le tableau source
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const double&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur le tableau des éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::AssignArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayDouble*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayDouble contenant les éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::AssignArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayDouble *src  =new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2015, MetaQuotes Software Corp.

```
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int     pos,          // Position
   double  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Update(int,double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,100.0))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer

shift

[in] La valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Shift(int,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Delete(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,     // Position du permier élément
   int  to        // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément du tableau à supprimer.

to

[in] Position du dernier élément du tableau à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::DeleteRange(int,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
double  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, DBL_MAX si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

DBL_MAX peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::At(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      double result=array.At(i);
      if(result==DBL_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  Erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const double&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::CompareArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const CArrayDouble*  src      // Pointeur vers la source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de la classe CArrayDouble contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::CompareArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Minimum

Retourne l'index du plus petit élément du tableau dans l'intervalle spécifié.

```
int  Minimum(
   int  start,      // index du début de l'intervalle
   int  count       // nombre d'éléments à traiter
   ) const
```

Paramètres

start

[in] Index de départ du tableau.

count

[in] Nombre d'éléments à traiter.

Valeur de retour

Index du plus petit élément du tableau dans l'intervalle spécifié.

© 2000-2015, MetaQuotes Software Corp.

Maximum

Retourne l'index du plus grand élément du tableau dans l'intervalle spécifié.

```
int  Maximum(
   int  start,      // index du début de l'intervalle
   int  count       // nombre d'éléments à traiter
   ) const
```

Paramètres

start

[in] Index de départ du tableau.

count

[in] Nombre d'éléments à traiter.

Valeur de retour

Index du plus grand élément du tableau dans l'intervalle spécifié.

© 2000-2015, MetaQuotes Software Corp.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   double  element      // Elément à insérer
   )
```

Paramètres

element

[in] valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::InsertSort(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(100.0))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Search(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search(100.0)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchGreat(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble:: SearchLess(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchGreatOrEqual(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchLessOrEqual(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchFirst(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchLast(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   double  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::SearchLinear(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear(100.0)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Save(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Load(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (87 pour CArrayDouble).

Exemple :

```
//--- exemple d'utilisation de CArrayDouble::Type()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CArrayString

La classe CArrayString est la classe des tableaux dynamiques de variables de type string.

Description

La classe CArrayString permet de travailler avec les tableaux dynamiques de variables de type string. La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CArrayString : public CArray
```

Titre

```
#include <Arrays\ArrayString.mqh>
```

Méthodes de Classe

Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Définit une nouvelle taille du tableau (plus petite)
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
AddArray	Ajoute des éléments d'un autre tableau à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée
AssignArray	Copie les éléments d'un autre tableau
AssignArray	Copie les éléments d'un autre tableau
Méthodes de mise à jour	

© 2000-2015, MetaQuotes Software Corp.

Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
CompareArray	Compare le tableau avec un autre tableau
Opérations sur les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Cherche un élément égal à celui donné dans un tableau trié
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal au modèle donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal au modèle donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle donné dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle donné dans un tableau trié
SearchLinear	Cherche l'élément égal à celui donné dans un tableau
Entrée/Sortie	
virtual Save	Sauvegarde le tableau de données dans un fichier
virtual Load	Charge les données du tableau depuis un fichier.

© 2000-2015, MetaQuotes Software Corp.

virtual Type	Retourne l'identifiant du type du tableau

© 2000-2015, MetaQuotes Software Corp.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Reserve(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réservation de la mémoire
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si la taille passée en argument est inférieure ou égale à 0.

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments situés sur la droite du tableau seront perdus. Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé en utilisant la pas définit grâce à la méthode Step (int), ou avec un pas de 16 sinon (valeur par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayString::Resize(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Shutdown()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   string  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter au tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Add(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(IntegerToString(i)))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const string&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur le tableau des éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::AddArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayString*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayString contenant les éléments source à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::AddArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- ajoute un autre tableau
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source
   delete src;
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   string  element,     // Elément à insérer
   int     pos          // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Insert(string,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(IntegerToString(i),0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const string&  src[],     // Tableau source
   int             pos        // Position
   )
```

Paramètres

src[]

[in] Référence sur un tableau d'éléments sources à insérer

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::InsertArray(const string &[],int)
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   CArrayString*  src,     // Pointeur vers la source
   int             pos      // Position
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayString contenant les éléments source à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::InsertArray(const CArrayString*,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- insére un autre tableau
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2015, MetaQuotes Software Corp.

```
      return;
     }
   //--- supprime le tableau source
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const string&  src[]      // Tableau source
   )
```

Paramètres

src[]

[in] Référence sur le tableau des éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::AssignArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayString*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayString contenant les éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::AssignArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayString *src  =new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source
```

© 2000-2015, MetaQuotes Software Corp.

```
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int     pos,          // Position
   string  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Update(int, string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,"ABC"))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer

shift

[in] La valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Shift(int,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Delete(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime l'élément
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,     // Position du permier élément
   int  to        // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément du tableau à supprimer.

to

[in] Position du dernier élément du tableau à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::DeleteRange(int,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
string  At(
   int  pos      // Position
   ) const
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, " (chaîne vide) si la position donnée n'existe pas (la dernière erreur est ERR_OUT_OF_RANGE).

Note

" (chaîne vide) peut bien sûr être une valeur valide pour un élément, il faut donc toujours vérifier le code de la dernière erreur si une valeur est retournée.

Exemple :

```
//--- exemple d'utilisation de CArrayString::At(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      string result=array.At(i);
      if(result=="" && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- Erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const string&  src[]      // Tableau source
   ) const
```

Paramètres

src[]

[in] Référence sur un tableau contenant les éléments sources à comparer.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayString::CompareArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArrays(
   const CArrayString*  src      // Pointeur vers la source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayString contenant les éléments source de comparaison.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayString::CompareArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- ajout les éléments du tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   string  element      // Elément à insérer
   )
```

Paramètres

element

[in] valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Exemple :

```
//--- exemple d'utilisation de CArrayString::InsertSort(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort("ABC"))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Search(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.Search("ABC")!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString::SearchGreat(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreat("ABC")!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString:: SearchLess(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLess("ABC")!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString:: SearchGreatOrEqual(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual("ABC")!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString:: SearchLessOrEqual(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLessOrEqual("ABC")!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString:: SearchFirst(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchFirst("ABC")!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayString:: SearchLast(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- recherche de l'élément
   if(array.SearchLast("ABC")!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLinear

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  SearchLinear(
   string  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Note

La méthode effectue un algorithme de recherche linéaire (ou recherche séquentielle) pour les tableaux non triés.

Exemple :

```
//--- exemple d'utilisation de CArrayString::SearchLinear(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLinear("ABC")!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Save(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      array.Add(IntegerToString(i));
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayString::Load(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilisation des éléments du tableau
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%s'",i,array.At(i));
```

© 2000-2015, MetaQuotes Software Corp.

```
      }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (89 pour CArrayString).

Exemple :

```
//--- exemple d'utilisation de CArrayString::Type()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CArrayObj

La classe CArrayObj est la classe des tableaux dynamiques de pointeurs vers des instances de CObject et de ses descendants.

Description

La classe CArrayObj permet de travailler avec un tableau dynamique de pointeurs vers des instances de CObject et de ses descendants. Ceci permet de travailler comme avec des tableaux de types de données primitifs à plusieurs dimensions.

La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans un tableau, dans un tableau trié, et la recherche dans un tableau trié. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Il y a quelques subtilités de la class CArrayObj.

Déclaration

```
class CArrayObj : public CArray
```

Titre

```
#include <Arrays\ArrayObj.mqh>
```

Méthode de Classe

Attributs	
FreeMode	Retourne le flag de gestion de la mémoire
FreeMode	Définit le flag de gestion de la mémoire
Contrôle de la mémoire	
Reserve	Alloue la mémoire pour augmenter la taille du tableau
Resize	Définit une nouvelle taille du tableau (plus petite)
Shutdown	Réinitialise le tableau et désalloue toute la mémoire.
Méthode d'ajout	
Add	Ajoute un élément à la fin du tableau
AddArray	Ajoute un élément à la fin du tableau
Insert	Insére un élément dans le tableau à la position spécifiée
InsertArray	Insére un tableau d'éléments à la position spécifiée

© 2000-2015, MetaQuotes Software Corp.

AssignArray	Copie les éléments d'un autre tableau
Méthodes de mise à jour	
Update	Met à jour l'élément situé à la position spécifiée
Shift	Déplace un élément du tableau depuis la position donnée et d'un décalage donné
Méthodes de suppression	
Detach	Retourne l'élément du tableau situé à la position spécifiée et l'enlève du tableau
Delete	Supprime l'élément du tableau situé à la position spécifiée
DeleteRange	Supprime un groupe d'éléments du tableau à partir de la position spécifiée
Clear	Supprime tous les éléments du tableau sans désallouer la mémoire
Méthodes d'accès	
At	Retourne l'élément du tableau situé à la position spécifiée
Méthodes de comparaison	
CompareArray	Compare le tableau avec un autre tableau
Opérations avec les tableaux triés	
InsertSort	Insére un élément dans un tableau trié
Search	Recherche un élément du tableau égal au modèle donné
SearchGreat	Cherche un élément supérieur à celui donné dans un tableau trié
SearchLess	Cherche un élément inférieur à celui donné dans un tableau trié
SearchGreatOrEqual	Cherche un élément supérieur ou égal à celui donné dans un tableau trié
SearchLessOrEqual	Cherche un élément inférieur ou égal à celui donné dans un tableau trié
SearchFirst	Cherche le premier élément égal au modèle dans un tableau trié
SearchLast	Cherche le dernier élément égal au modèle dans un tableau trié
Entrée/Sortie	
Save	Sauvegarde le tableau de données dans un

© 2000-2015, MetaQuotes Software Corp.

	fichier
Load	Charge les données du tableau depuis un fichier.
Type	Retourne l'identifiant du type du tableau

Classes dérivées :

- CIndicator
- CIndicators

Une application pratique des tableaux sont les descendants de la classe CObject (incluant toutes les classes de la bibliothèque standard).

Par exemple, considérons un tableau à 2 dimensions :

```
#include <Arrays\ArrayDouble.mqh>
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int i,j;
   int first_size=10;
   int second_size=100;
//--- création du tableau
   CArrayObj    *array=new CArrayObj;
   CArrayDouble *sub_array;
//---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
//--- création des sous-tableaux
   for(i=0;i<first_size;i++)
     {
      sub_array=new CArrayDouble;
      if(sub_array==NULL)
        {
         delete array;
         printf("Object create error");
         return;
        }
      //--- remplissage du tableau
      for(j=0;j<second_size;j++)
        {
         sub_array.Add(i*j);
        }
      array.Add(sub_array);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- tableau créé avec succès
   for(i=0;i<first_size;i++)
     {
      sub_array=array.At(i);
      for(j=0;j<second_size;j++)
        {
         double element=sub_array.At(j);
         //--- utilisation de l'élément du tableau
        }
     }
   delete array;
  }
```

Subtilités

La classe possède un mécanisme de contrôle dynamique de la mémoire, veillez donc à faire attention lorsque vous travaillez avec des éléments dans un tableau.

Le mécanisme de gestion de la mémoire peut être activé ou désactivé en utilisant la méthode FreeMode (bool). Par défaut, le mécanisme est activé.

De la même façon, il y a deux options de gestion de la classe CArrayObj :

1. Mécanisme de gestion de la mémoire activé. (par défaut)

Dans ce cas, la classe CArrayObj prend la responsabilité de libérer les éléments de la mémoire après leur suppression du tableau. Dans ce mode, l'utilisateur ne devrait pas libérer les éléments d'un tableau.

Exemple :

```
   int i;
//--- Création d'un tableau
   CArrayObj *array=new CArrayObj;
//--- Remplissage des éléments du tableau
   for(i=0;i<10;i++) array.Add(new CObject);
//--- Fait quelque chose
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- Action sur un élément
      . . .
     }
//--- Supprime le tableau et ses éléments
   delete array;
```

2. Mécanisme de gestion de la mémoire désactivé.

Dans ce cas, la classe CArrayObj ne prend pas la responsabilité de libérer les éléments de la mémoire après leur suppression du tableau. Dans ce mode, l'utilisateur doit libérer les éléments d'un tableau.

© 2000-2015, MetaQuotes Software Corp.

Exemple :

```
   int i;
//--- Création d'un tableau
   CArrayObj *array=new CArrayObj;
//--- Désactive le mécanisme de la gestion de la mémoire
   array.FreeMode(false);
//--- Remplissage des éléments du tableau
   for(i=0;i<10;i++) array.Add(new CObject);
//--- Fait quelque chose
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- Action sur un élément
      . . .
     }
//--- Supprime les éléments du tableau
   while(array.Total()) delete array.Detach();
//--- Supprime le tableau vide
   delete array;
```

© 2000-2015, MetaQuotes Software Corp.

FreeMode

Retourne le mode de gestion de la mémoire.

```
bool  FreeMode() const
```

Valeur de Retour

Mode de gestion de la mémoire.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::FreeMode()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le mode de gestion de la mémoire
   bool array_free_mode=array.FreeMode();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

FreeMode

Définit le mode de gestion de la mémoire.

```
void  FreeMode(
   bool  mode      // Nouveau mode
   )
```

Paramètres

mode

[in] Nouveau mode de gestion de la mémoire.

Valeur de Retour

Aucune.

Note

La définition du mode de gestion de la mémoire est un point important de l'utilisation de la classe CArrayObj. Puisque les éléments du tableau sont des pointeurs vers des objets, il est important de savoir comment les gérer lorsqu'ils sont enlevés du tableau.

Si le mode est activé, enlever un élément du tableau déclenche sa suppression par un appel à l'opérateur delete. si le mode n'est pas activé, le pointeur vers l'objet existe toujours quelque part dans le programme et doit être libéré par le programme.

Si le mode de gestion de la mémoire est réinitialisé, le programmeur doit prendre la responsabilité de la suppression des éléments de la liste avant la fin du programme pour libérer la mémoire utilisée lors de la création de l'élément avec l'opérateur new. Avec de gros volumes de données, ceci peut empêcher votre terminal de fonctionner.

Avec de gros volumes de données, cela peut amener à des problèmes d'utilisation du terminal. Il y a également un autre risque.

Avec les listes de pointeurs, les éléments sont stockés sous forme de variables locales. Après la suppression de la liste, ne pas supprimer ces variables peut provoquer des erreurs critiques et des crashs du programme. Par défaut, le mode de gestion de la mémoire est activé, la classe de la liste est responsable de la libération de la mémoire de ses éléments.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::FreeMode(bool)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- réinitialise le flag du mode de suppression
   array.FreeMode(false);
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Reserve

Alloue la mémoire pour augmenter la taille du tableau.

```
bool  Reserve(
   int  size      // Nombre d'éléments à réserver
   )
```

Paramètres

size

[in] Le nombre d'éléments supplémentaires du tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre d'éléments à réserver est inférieur ou égal à 0 ou si la taille du tableau n'a pas pu être augmentée.

Note

Pour réduire la fragmentation de la mémoire, le changement de la taille du tableau est réalisé avec le pas définit précédemment grâce à la méthode Step (int), ou avec un pas de 16 sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Reserve(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Resize

Définit une nouvelle taille du tableau (plus petite).

```
bool  Resize(
   int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du tableau.

Valeur de Retour

Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé avec le pas définit avec la méthode Step (int), ou sinon avec un pas de 16 (par défaut).

Note

Changer la taille du tableau permet d'optimiser l'utilisation de la mémoire. Les éléments en excés situés sur la droite sont perdus. La mémoire des éléments perdus est libérée ou pas suivant le mode de gestion de la mémoire.

Pour réduire la fragmentation de la mémoire, le changement de taille du tableau est réalisé avec le pas définit avec la méthode Step (int), ou sinon avec un pas de 16 (par défaut).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Resize(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- redimmensionne le tableau
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
```

© 2000-2015, MetaQuotes Software Corp.

```
}
```

© 2000-2015, MetaQuotes Software Corp.

Clear

Supprime tous les éléments du tableau sans désallouer la mémoire

```
void  Clear()
```

Valeur de Retour

Aucune.

Note

Si le mode de gestion de la mémoire est activé, les éléments supprimés sont exemptés.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Clear()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- vide le tableau
   array.Clear();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shutdown

Réinitialise le tableau et désalloue toute la mémoire.

```
bool  Shutdown()
```

Valeur de Retour

vrai si réalisé avec succès, faux si une erreur est survenue.

Note

Si le mode de gestion de la mémoire est activé, les éléments supprimés sont exemptés.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Shutdown()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- détruit le tableau
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CreateElement

Crée un nouvel élément du tableau à la position spécifiée.

```
bool  CreateElement(
   int  index      // Position
   )
```

Paramètres

index

[in] Position d'insertion de l'élément dans le tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être créé.

Note

La méthode CreateElement (int) de la classe CArrayObj retourne toujours faux et n'exécute aucune action. La méthode CreateElement (int) devrait être implémenté si nécessaire dans les classes dérivées.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::CreateElement(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         size=100;
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- remplissage du tableau
   array.Reserve(size);
   for(int i=0;i<size;i++)
     {
      if(!array.CreateElement(i))
        {
         printf("Element create error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
```

© 2000-2015, MetaQuotes Software Corp.

```
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Add

Ajoute un élément à la fin du tableau.

```
bool  Add(
   CObject*  element      // Elément à ajouter
   )
```

Paramètres

element

[in] valeur de l'élément à ajouter au tableau.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être ajouté.

Note

L'élément n'est pas ajouté au tableau si le pointeur est invalide (NULL par exemple).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Add(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute 100 éléments dans le tableau
   for(int i=0;i<100;i++)
     {
      if(!array.Add(new CObject))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AddArray

Ajoute des éléments d'un autre tableau à la fin du tableau.

```
bool  AddArray(
   const CArrayObj *  src      // Pointeur vers le tableau source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayDouble contenant les éléments sources à ajouter.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Note

L'ajout des éléments dans le tableau est en fait une copie de pointeurs. En appelant la méthode, plusieurs variables peuvent donc contenir un pointeur vers un objet dynamique.

```
//--- exemple
extern bool       make_error;
extern int        error;
extern CArrayObj *src;
//--- Crée une nouvelle instance de CArrayObj
//--- La gestion de la mémoire est par défaut activée
CArrayObj *array=new CArrayObj;
//--- Ajoute (copie) les éléments du tableau source
if(array!=NULL)
   bool result=array.AddArray(src);
if(make_error)
  {
   //--- Effectue une action déclenchant une erreur
   switch(error)
     {
      case 0:
         //--- Enlève le tableau source sans vérifier son mode de gestion de la mémoi
         delete src;
         //--- Résultat :
         //--- Il est possible de référencer un élément par un pointeur invalide dans
         break;
      case 1:
         //--- Désactive le mode de gestion de la mémoire dans le tableau source
         if(src.FreeMode()) src.FreeMode(false);
         //--- Mais ne supprime pas le tableau source
         //--- Résultat :
         //--- Après avoir supprimé le tableau destination, il est possible de référer
         break;
      case 2:
```

© 2000-2015, MetaQuotes Software Corp.

```
         //--- Désactive le mode de gestion de la mémoire dans le tableau source
         src.FreeMode(false);
         //--- Désactive le mode de gestion de la mémoire dans le tableau destination
         array.FreeMode(false);
         //--- Résultat :
         //--- Après la fin du programme, une "fuite mémoire" apparaît
         break;
      }
   }
else
   {
    //--- Désactive le mode de gestion de la mémoire dans le tableau source
    if(src.FreeMode()) src.FreeMode(false);
    //--- Supprime le tableau source
    delete src;
    //--- Résultat :
    //--- Adresser un élément du tableau source sera correct
    //--- La suppression du tableau destination supprimera tous ses éléments
   }
```

Exemple :

```
//--- exemple d'utilisation de CArrayObj::AddArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
   {
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
      {
       printf("Object create error");
       return;
      }
    //--- crée le tableau source
    CArrayObj *src=new CArrayObj;
    if(src==NULL)
      {
       printf("Object create error");
       delete array;
       return;
      }
    //--- réinitialise le flag du mode de suppression
    src.FreeMode(false);
    //--- remplit le tableau source
    //--- . . .
    //--- ajoute un autre tableau
    if(!array.AddArray(src))
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source sans les éléments
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Insert

Insére un élément dans le tableau à la position spécifiée.

```
bool  Insert(
   CObject*  element,     // Elément à insérer
   int       pos          // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans le tableau

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Note

L'élément n'est pas ajouté au tableau si le pointeur est invalide (NULL par exemple).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Insert(CObject*,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére les éléments
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(new CObject,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
```

© 2000-2015, MetaQuotes Software Corp.

```
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertArray

Insére un tableau d'éléments à la position spécifiée.

```
bool  InsertArray(
   const CArrayObj*  src,     // Pointeur vers la source
   int               pos      // Position
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayObj contenant les éléments à insérer.

pos

[in] Position d'insertion de l'élément dans le tableau

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être ajoutés.

Note

Voir également : CArrayObj::AddArray(const CArrayObj*).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::InsertArray(const CArrayObj*,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- réinitialise le flag du mode de suppression
   src.FreeMode(false);
   //--- remplit le tableau source
   //--- . . .
   //--- insére un autre tableau
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
      return;
     }
   //--- supprime le tableau source sans les éléments
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

AssignArray

Copie les éléments d'un autre tableau.

```
bool  AssignArray(
   const CArrayObj*  src      // Pointeur vers la source
   )
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayObj contenant les éléments source à copier.

Valeur de Retour

vrai si réalisé avec succès, faux si les élément ne peuvent pas être copiés.

Note

Si le tableau destination de la fonction AssignArray n'est pas vide, tous ses éléments seront supprimés du tableau, et si le mode de gestion de la mémoire est activé, la mémoire sera libérée. Le tableau destination est la copie exacte du tableau source. Voir éfalement CArrayObj::AddArray(const CArrayObj*).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::AssignArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- réinitialise le flag du mode de suppression
   src.FreeMode(false);
   //--- remplit le tableau source
   //--- . . .
   //--- assigne un autre tableau
   if(!array.AssignArray(src))
```

© 2000-2015, MetaQuotes Software Corp.

```
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- les tableaux sont identiques
   //--- supprime le tableau source sans les éléments
   delete src;
   //--- utilise le tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Update

Met à jour l'élément situé à la position spécifiée

```
bool  Update(
   int       pos,          // Position
   CObject*  element       // Valeur
   )
```

Paramètres

pos

[in] Position de l'élément du tableau à mettre à jour

element

[in] Nouvelle valeur de l'élément

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être modifié.

Note

L'élément n'est pas modifié si le pointeur passé en paramètre est invalide (NULL par exemple). Si la mode de gestion de la mémoire est activé, la mémoire est libérée si besoin.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Update(int,CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- met l'élément à jour
   if(!array.Update(0,new CObject))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Shift

Déplace un élément du tableau depuis la position donnée et d'un décalage donné.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Décalage
   )
```

Paramètres

pos

[in] Position de l'élément à déplacer

shift

[in] La valeur de déplacement (positif ou négatif).

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Shift(int,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- décale l'élément
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Detach

Détache l'élément du tableau situé à la position spécifiée.

```
CObject*  Detach(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

Pointeur sur l'objet détaché en cas de succès, NULL si l'élément ne peut pas être détaché.

Note

Lorsqu'un élément du tableau est détaché, il n'est pas supprimé, quelque soit le mode de gestion de la mémoire. Le pointeur vers l'élément du tableau est retiré de la liste des éléments à libérer.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Detach(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   CObject *object=array.Detach(0);
   if(object==NULL)
     {
      printf("Detach error");
      delete array;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- supprime l'élément
   delete object;
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime l'élément du tableau situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans le tableau de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Note

Si le mode de gestion de la mémoire est activé, les éléments supprimés sont exemptés.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Delete(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteRange

Supprime un groupe d'éléments du tableau à partir de la position spécifiée.

```
bool  DeleteRange(
   int  from,     // Position du permier élément
   int  to        // Position du dernier élément
   )
```

Paramètres

from

[in] Position du premier élément du tableau à supprimer.

to

[in] Position du dernier élément du tableau à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être supprimés.

Note

Si le mode de gestion de la mémoire est activé, les éléments supprimés sont exemptés.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::DeleteRange(int,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- supprime les éléments
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du tableau situé à la position spécifiée.

```
CObject*  At(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position de l'élément désiré dans le tableau.

Valeur de Retour

La valeur de l'élément en cas de succès, NULL si la position donnée n'existe pas.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::At(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      CObject *result=array.At(i);
      if(result==NULL)
        {
         //--- Erreur de lecture du tableau
         printf("Get element error");
         delete array;
         return;
        }
      //--- utilisation de l'élément
      //--- . . .
     }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareArray

Compare le tableau avec un autre tableau.

```
bool  CompareArray(
   const CArrayObj*  src      // Pointeur vers la source
   ) const
```

Paramètres

src

[in] Pointeur vers une instance de classe CArrayObj contenant les éléments source de comparaison.

Valeur de Retour

vrai si les tableaux sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::CompareArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée le tableau source
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- remplit le tableau source
   //--- . . .
   //--- comparaison avec un autre tableau
   int result=array.CompareArray(src);
   //--- supprime les tableaux
   delete src;
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

InsertSort

Insére un élément dans un tableau trié.

```
bool  InsertSort(
   CObject*  element      // Elément à insérer
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans un tableau trié

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être inséré.

Note

L'élément n'est pas ajouté au tableau si le pointeur est invalide (NULL par exemple).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::InsertSort(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- insérer l'élément
   if(!array.InsertSort(new CObject))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Search

Cherche un élément égal au modèle donné dans un tableau trié.

```
int  Search(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Search(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.Search(sample)!=-1) printf("Element found");
   else                         printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreat

Cherche un élément supérieur à celui donné dans un tableau trié.

```
int  SearchGreat(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::SearchGreat(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchGreat(sample)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLess

Cherche un élément inférieur à celui donné dans un tableau trié.

```
int  SearchLess(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj:: SearchLess(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLess(sample)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchGreatOrEqual

Cherche un élément supérieur ou égal au modèle donné dans un tableau trié.

```
int  SearchGreatOrEqual(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::SearchGreatOrEqual(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchGreatOrEqual(sample)!=-1) printf("Element found");
   else                                     printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLessOrEqual

Cherche un élément inférieur ou égal au modèle donné dans un tableau trié.

```
int  SearchLessOrEqual(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj:: SearchLessOrEqual(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLessOrEqual(sample)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchFirst

Cherche le premier élément égal au modèle donné dans un tableau trié.

```
int  SearchFirst(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::SearchFirst(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchFirst(sample)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SearchLast

Trouve le dernier élément égal au modèle dans un tableau trié.

```
int  SearchLast(
   CObject*  element      // Modèle
   ) const
```

Paramètres

element

[in] L'élément à rechercher dans le tableau.

Valeur de Retour

La position de l'élément trouvé en cas de succès, -1 si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CArrayObj:: SearchLast(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- trie le tableau
   array.Sort();
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- définition des attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(array.SearchLast(sample)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde le tableau de données dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Save(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments au tableau
   //--- . . .
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les données du tableau depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] Handle du fichier précédemment ouvert (en mode binaire) avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

Lors de la lecture du fichier, chaque élément du tableau est créé en appelant la méthode CArrayObj::CreateElement(int).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Load(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- utilisation des éléments du tableau
   //--- . . .
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type du tableau

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du tableau (7778 pour CArrayObj).

Exemple :

```
//--- exemple d'utilisation de CArrayObj::Type()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type du tableau
   int type=array.Type();
   //--- supprime le tableau
   delete array;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CList

La classe CList est une liste d'instance de classes CObject et de ses descendants.

Description

La classe CList fournit la possibilité de travailler avec une liste d'instance de CObject et de ses descendants La classe implémente les fonctions d'ajout, d'insertion et de suppression des éléments dans la liste, de tri et de recherche dans la liste triée. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Il y a quelques subtilités à connaître lorsque l'on travaille avec la classe CList. La classe possède un mécanisme de contrôle dynamique de la mémoire, veillez donc à faire attention lorsque vous travaillez avec des éléments dans la liste.

Les subtilités du mécanisme de gestion de la mémoire sont similaires à celles décrites pour la classe CArrayObj.

Déclaration

```
class CList : public CObject
```

Titre

```
#include <Arrays\List.mqh>
```

Méthodes de Classe

Attributs	
FreeMode	Retourne le mode de gestion de la mémoire lors de la suppression d'éléments de la liste.
FreeMode	Retourne le mode de gestion de la mémoire lors de la suppression d'éléments de la liste.
Total	Retourne le nombre d'éléments dans la liste
IsSorted	Retourne le flag permettant de savoir si la liste est triée ou pas
SortMode	Retourne le mode de tri
Méthodes de création	
CreateElement	Crée un nouvel élément dans la liste
Méthode d'ajout	
Add	Ajoute un élément à la fin de la liste
Insert	Insére un élément dans la liste à la position spécifiée
Méthodes de suppression	

© 2000-2015, MetaQuotes Software Corp.

DetachCurrent	Détache un élément de la liste situé à la position spécifiée sans le supprimer "physiquement"
DeleteCurrent	Supprime l'élément courant de la liste
Delete	Supprime l'élément de la liste situé à la position spécifiée
Clear	Supprime tous les éléments de la liste
Navigation	
IndexOf	Retourne la position de l'élément dans la liste
GetNodeAtIndex	Retourne l'élément de la liste correspondant à la position spécifiée
GetFirstNode	Retourne le premier élément de la liste
GetPrevNode	Retourne l'élément précédent dans la liste
GetCurrentNode	Retourne l'élément courant de la liste
GetNextNode	Retourne le prochain élément de la liste
GetLastNode	Retourne le dernier élément
Méthodes de tri	
Sort	Trie la liste
MoveToIndex	Déplace l'élément courant de la liste à la position spécifiée
Exchange	Intervertit deux éléments de la liste
Méthodes de comparaison	
CompareList	Compare la liste avec une autre liste
Méthodes de recherche	
Search	Cherche un élément égal au modèle donné dans une liste triée
Entrée/Sortie	
virtual Save	Sauvegarde les données de la liste dans un fichier.
virtual Load	Charge les données de la liste depuis un fichier
virtual Type	Retourne l'identifiant du type de la liste.

© 2000-2015, MetaQuotes Software Corp.

FreeMode

Retourne le mode de gestion de la mémoire lors de la suppression d'éléments de la liste.

```
bool  FreeMode() const
```

Valeur de Retour

Mode de gestion de la mémoire.

Exemple :

```
//--- exemple d'utilisation de CList::FreeMode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le mode de gestion de la mémoire
   bool list_free_mode=list.FreeMode();
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

FreeMode

Définit le mode de gestion de la mémoire lors de la suppression d'éléments de la listel.

```
void  FreeMode(
   bool  mode      // Nouvelle valeur
   )
```

Paramètres

mode

[in] Nouveau mode de gestion de la mémoire.

Note

La gestion de la mémoire est un élément important de la classe CList. Puisque les éléments de la liste sont des pointerus vers des objets, il est important de savoir quoi en faire lorsqu'il est retiré de la liste. Si le mode de gestion de la mémoire est activé, l'élément enlevé de la liste est alors automatiquement supprimé par un appel à l'opérateur delete. Si le mode de gestion de la mémoire n'est pas activé, le pointeur existe toujours lorsque l'élément est enlevé de la liste et c'est au programme de se charger de le supprimer et de libérer la mémoire.

Si le mode de gestion de la mémoire est réinitialisé, le programmeur doit prendre la responsabilité de la suppression des éléments de la liste avant la fin du programme pour libérer la mémoire utilisée lors de la création de l'élément avec l'opérateur new. Avec de gros volumes de données, ceci peut empêcher votre terminal de fonctionner.

Si le mode de gestion de la mémoire n'est pas réinitialisé, un autre risque subsiste. Avec les listes de pointeurs, les éléments sont stockés sous forme de variables locales. Après la suppression de la liste, ne pas supprimer ces variables peut provoquer des erreurs critiques et des crashs du programme. Par défaut, le mode de gestion de la mémoire est activé, la classe de la liste est responsable de la libération de la mémoire de ses éléments.

Exemple :

```
//--- exemple d'utilisation de CList::FreeMode(bool)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- réinitialise le flag du mode de suppression
   list.FreeMode(false);
   //--- utilisation de la liste
   //--- . . .
   //--- supprime la liste
```

© 2000-2015, MetaQuotes Software Corp.

```
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Total

Retourne le nombre d'éléments dans la liste.

```
int  Total() const
```

Valeur de Retour

Nombre d'éléments dans la liste.

Exemple :

```
//--- exemple d'utilisation de CList::Total()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie le nombre total
   int total=list.Total();
   //--- utilisation de la liste
   //--- ...
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

IsSorted

Retourne le flag permettant de savoir si la liste est triée ou pas

```
bool  IsSorted(
   int  mode=0      // Mode de tri
   ) const
```

Paramètres

mode=0

[in] Mode de tri à vérifier

Valeur de Retour

Flag de la liste triée. Si la liste est triée suivant le mode de tri spécifié? vrai, sinon? faux.

Note

Le mode de tri d'une liste ne peut pas être changé directement. Le mode est défini avec la méthode Sort (int) et réinitialise toutes les méthodes d'ajout / insertion.

Exemple :

```
//--- exemple d'utilisation de CList::IsSorted()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie le mode de tri
   if(list.IsSorted(0))
     {
      //--- utilisation de méthodes sur les listes triées
      //--- ...
     }
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

SortMode

Retourne le mode de tri.

```
int  SortMode() const
```

Valeur de Retour

Mode de tri, ou -1 si la liste n'est pas triée.

Exemple :

```
//--- exemple d'utilisation de CList::SortMode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- vérifie le mode de tri
   int sort_mode=list.SortMode();
   //--- utilisation de la liste
   //--- ...
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CreateElement

Crée un nouvel élément dans la liste.

```
CObject*  CreateElement()
```

Valeur de Retour

Pointeur sur le nouvel élément créé, NULL si l'élément ne peut pas être créé.

Note

La méthode CreateElement (int) de la classe CList retourne toujours NULL et n'exécute aucune action. La méthode CreateElement () devrait être implémenté si nécessaire dans les classes dérivées.

Exemple :

```
//--- exemple d'utilisation de CList::CreateElement(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    size=100;
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- remplissage de la liste
   for(int i=0;i<size;i++)
     {
      CObject *object=list.CreateElement();
      if(object==NULL)
        {
         printf("Element create error");
         delete list;
         return;
        }
      list.Add(object);
     }
   //--- utilisation de la liste
   //--- . . .
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Add

Ajoute un élément à la fin de la liste.

```
int  Add(
   CObject*  element      // Elément à ajouter
   )
```

Paramètres

element

[in] Valeur de l'élément à ajouter à la liste.

Valeur de Retour

La position de l'élément ajouté en cas de succès, -1 en cas d'erreur.

Note

L'élément n'est pas ajouté à la liste si le pointeur passé en paramètre est invalide (NULL par exemple).

Exemple :

```
//--- exemple d'utilisation de CList::Add(Cobject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoue 100 éléments
   for(int i=0;i<100;i++)
     {
      if(list.Add(new CObject)==-1)
        {
         printf("Element addition error");
         delete list;
         return;
        }
     }
   //--- utilisation de la liste
   //--- . . .
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Insert

Insére un élément dans la liste à la position spécifiée.

```
int  Insert(
   CObject*  element,     // Elément à insérer
   int       pos          // Position
   )
```

Paramètres

element

[in] Valeur de l'élément à insérer dans la liste.

pos

[in] Position d'insertion de l'élément dans la liste

Valeur de Retour

La position de l'élément inséré en cas de succès, -1 en cas d'erreur.

Note

L'élément n'est pas ajouté à la liste si le pointeur passé en paramètre est invalide (NULL par exemple).

Exemple :

```
//--- exemple d'utilisation de CList::Insert(CObject*,int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insére 100 éléments
   for(int i=0;i<100;i++)
     {
      if(list.Insert(new CObject,0)==-1)
        {
         printf("Element insert error");
         delete list;
         return;
        }
     }
   //--- utilisation de la liste
   //--- . . .
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

DetachCurrent

Détache l'élément de la liste situé à la position spécifiée sans le supprimer "physiquement".

```
CObject*  DetachCurrent()
```

Valeur de Retour

Pointeur sur l'objet détaché en cas de succès, NULL si l'élément ne peut pas être détaché.

Note

Lorsqu'un élément de la liste est détaché, il n'est pas supprimé, quelque soit le mode de gestion de la mémoire. Le pointeur vers l'élément de la liste est retiré de la liste des éléments à libérer.

Exemple :

```
//--- exemple d'utilisation de CList::DetachCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.DetachCurrent();
   if(object==NULL)
     {
      printf("Detach error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- supprime l'élément
   delete object;
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteCurrent

Supprime l'élément courant de la liste.

```
bool  DeleteCurrent()
```

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Note

Si la mode de gestion de la mémoire est activé, la mémoire est libérée.

Exemple :

```
//--- exemple d'utilisation de CList::DeleteCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   if(!list.DeleteCurrent())
     {
      printf("Delete error");
      delete list;
      return;
     }
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime l'élément de la liste situé à la position spécifiée.

```
bool  Delete(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position dans la liste de l'élément à supprimer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être supprimé.

Note

Si la mode de gestion de la mémoire est activé, la mémoire est libérée.

Exemple :

```
//--- exemple d'utilisation de CList::Delete(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   if(!list.Delete(0))
     {
      printf("Delete error");
      delete list;
      return;
     }
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Clear

Supprime tous les éléments de la liste.

```
void  Clear()
```

Note

Si le mode de gestion de la mémoire est activé, les éléments supprimés sont exemptés.

Exemple :

```
//--- exemple d'utilisation de CList::Clear()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   //--- vide la liste
   list.Clear();
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

IndexOf

Retourne la position de l'élément dans la liste

```
int  IndexOf(
   CObject*  element      // Pointeur vers l'élément
   )
```

Paramètres

element

[in] Pointeur vers l'élément de la liste.

Valeur de Retour

Position de l'élément dans la liste ou -1.

Exemple :

```
//--- exemple d'utilisation de CList::IndexOf(CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   CObject *object=new CObject;
   if(object==NULL)
     {
      printf("Element create error");
      delete list;
      return;
     }
   if(list.Add(object))
     {
      int pos=list.IndexOf(object);
     }
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

GetNodeAtIndex

Retourne l'élément de la liste correspondant à la position spécifiée.

```
CObject*  GetNodeAtIndex(
   int  pos      // position
   )
```

Paramètres

pos

[in] Position de l'élément.

Valeur de retour

Pointeur sur l'élément en cas de succès, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetNodeAtIndex(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetNodeAtIndex(10);
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

GetFirstNode

Retourne le premier élément de la liste.

```
CObject*  GetFirstNode()
```

Valeur de Retour

Pointeur sur le premier élément, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetFirstNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetFirstNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

GetPrevNode

Retourne l'élément précédent de la liste.

```
CObject*  GetPrevNode()
```

Valeur de Retour

Pointeur sur l'élément précédent de la liste, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetPrevNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetPrevNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

GetCurrentNode

Retourne l'élément courant de la liste

```
CObject*  GetCurrentNode()
```

Valeur de Retour

Pointeur sur l'élément courant, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetCurrentNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetCurrentNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

GetNextNode

Retourne le prochain élément de la liste

```
CObject*  GetNextNode()
```

Valeur de Retour

Pointeur sur le prochain élément en cas de succès, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetNextNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetNextNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

GetLastNode

Retourne le dernier élément de la liste.

```
CObject*  GetLastNode()
```

Valeur de Retour

Pointeur sur le dernier élément, NULL si un pointeur ne peut pas être retourné.

Exemple :

```
//--- exemple d'utilisation de CList::GetLastNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   CObject *object=list.GetLastNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- utilisation de l'élément
   //--- . . .
   //--- ne pas supprimer l'élément
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Sort

Trie une liste.

```
void  Sort(
   int  mode      // Mode de tri
   )
```

Paramètres

mode

[in] Mode de tri.

Valeur de Retour

Aucune.

Note

Le tri de la liste est toujours effectué en mode ascendant.

Exemple :

```
//--- exemple d'utilisation de CList::Sort(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- Trie avec le mode 0
   list.Sort(0);
   //--- utilisation de la liste
   //--- ...
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

MoveToIndex

Déplace l'élément courant de la liste à la position spécifiée.

```
bool  MoveToIndex(
   int  pos      // Position
   )
```

Paramètres

pos

[in] Position de l'élément de la liste à déplacer.

Valeur de Retour

vrai si réalisé avec succès, faux si l'élément ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CList::MoveToIndex(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- déplacer le noeud courant pour commencer
   list.MoveToIndex(0);
   //--- utilisation de la liste
   //--- . . .
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Exchange

Intervertit deux éléments de la liste.

```
bool  Exchange(
   CObject*  node1,     // Premier élément de la liste
   CObject*  node2      // Deuxième élément de la liste
   )
```

Paramètres

node1

[in] Elément de la liste

node2

[in] Elément de la liste

Valeur de Retour

vrai si réalisé avec succès, faux si les éléments ne peuvent pas être intervertis.

Exemple :

```
//--- exemple d'utilisation de CList::Exchange(CObject*,CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- déplacement des éléments
   list.Exchange(list.GetFirstNode(),list.GetLastNode());
   //--- utilisation de la liste
   //--- . . .
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CompareList

Compare la liste avec une autre liste.

```
bool  CompareList(
   CList*  list      // Liste de comparaison
   )
```

Paramètres

list

[in] Pointeur vers une instance de la classe CList contenant les éléments sources à comparer.

Valeur de Retour

vrai si les listes sont identiques, faux sinon.

Exemple :

```
//--- exemple d'utilisation de CList::CompareList(const CList*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- crée la liste source
   CList *src=new CList;
   if(src==NULL)
     {
      printf("Object create error");
      delete list;
      return;
     }
   //--- remplissage de la liste source
   //--- . . .
   //--- comparaison avec une autre liste
   bool result=list.CompareList(src);
   //--- supprime la listes
   delete src;
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Search

Cherche un élément égal au modèle donné dans une liste triée.

```
CObject*  Search(
   CObject*  element      // Modèle
   )
```

Paramètres

element

[in] Modèle à rechercher dans la liste.

Valeur de Retour

Pointeur sur l'élément trouvé en cas de succès, NULL si l'élément n'a pas été trouvé.

Exemple :

```
//--- exemple d'utilisation de CList::Search(CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   //--- trie la liste
   list.Sort(0);
   //--- création du modèle
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete list;
      return;
     }
   //--- définit les attributs du modèle
   //--- . . .
   //--- recherche de l'élément
   if(list.Search(sample)!=NULL) printf("Element found");
   else                          printf("Element not found");
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les données de la liste dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier précédemment ouvert avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CList::Save(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ajoute des éléments dans la liste
   //--- . . .
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- supprime la liste
   delete list;
```

© 2000-2015, MetaQuotes Software Corp.

```
}
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les données de la liste depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier précédemment ouvert avec la fonction FileOpen (...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

Lors de la lecture du fichier, chaque élément de la liste est créé en appelant la méthode CList::CreateElement().

Exemple :

```
//--- exemple d'utilisation de CLoad::Load(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
   //--- utilisation des éléments de la liste
   //--- . . .
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de la liste.

```
virtual int  Type()
```

Valeur de Retour

Identifant du type de la liste (7779 pour CList).

Exemple :

```
//--- exemple d'utilisation de CList::Type()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- récupère le type de la liste
   int type=list.Type();
   //--- supprime la liste
   delete list;
  }
```

© 2000-2015, MetaQuotes Software Corp.

CTreeNode

La classe CTreeNode représente un noeud de l'arbre binaire CTree.

Description

La classe CTreeNode permet de travailler avec les noeuds d'un arbre binaire CTree. Les options de navigation au travers de l'arbre sont implémentées dans la classe. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Déclaration

```
class CTreeNode : public CObject
```

Titre

```
#include <Arrays\TreeNode.mqh>
```

Méthodes de Classe

Attributs	
Owner	Retourne/Définit le pointeur du noeud père
Left	Retourne/Définit le pointeur situé à gauche
Right	Retourne/Définit le pointeur situé à droite
Balance	Retourne la balance du noeud
BalanceL	Retourne la balance de la sous-branche de gauche du noeud
BalanceR	Retourne la balance de la sous-branche de droite du noeud
Création d'u nouvel élément	
CreateSample	Création d'un nouveau noeud
Comparaison	
RefreshBalance	Recalcule la balance du noeud
Recherche	
GetNext	Retourne le pointeur du noeud suivant
Entrée/Sortie	
SaveNode	Sauvegarde les données du noeud dans un fichier
LoadNode	Charge les données d'un noeud depuis un fichier
virtual Type	Retourne l'identifiant du type du noeud

© 2000-2015, MetaQuotes Software Corp.

Classes dérivées :

- CTree

Les arbres de classes dérivant de CTreeNode sont mis en application.

Une classe dérivant de la classe doit implémenter les méthodes : CreateSample pour créer une nouvelle instance de la classe dérivant de la classe CTreeNode, Compare pour comparer les valeurs des champs clés de la classe dérivant de la classe CTreeNode, Type (si nécéssité d'identifier un noeud), SaveNode et LoadNode (si nécéssité de travailler avec des fichiers).

Considérons l'exemple d'une classe dérivée de la classe CTree.

```
//+------------------------------------------------------------------+
//|                                                   MyTreeNode.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                              http://www.metaquotes.net/ |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\TreeNode.mqh>
//+------------------------------------------------------------------+
//| Décrit la classe dérivée de CTreeNode. |
//+------------------------------------------------------------------+
//| Class CMyTreeNode. |
//| But : Classe d'un élément d'un arbre binaire. |
//|             Descendant de la classe CTreeNode. |
//+------------------------------------------------------------------+
class CMyTreeNode : public CTreeNode
  {
protected:
   //--- données utlisateur
   long              m_long;           // champ clé de type long
   double            m_double;         // variable personnelle de type double
   string            m_string;         // variable personnelle de type string
   datetime          m_datetime;       // variable personnelle de type datetime

public:
                     CMyTreeNode();
   //--- méthodes d'accès aux données utilisateur
   long              GetLong(void)               { return(m_long); }
   void              SetLong(long value)         { m_long=value;  }
   double            GetDouble(void)             { return(m_double); }
   void              SetDouble(double value)     { m_double=value;  }
   string            GetString(void)             { return(m_string); }
   void              SetString(string value)     { m_string=value;  }
   datetime          GetDateTime(void)           { return(m_datetime); }
   void              SetDateTime(datetime value) { m_datetime=value;  }
   //--- méthodes de travail avec les fichiers
```

© 2000-2015, MetaQuotes Software Corp.

```
   virtual bool      Save(int file_handle);
   virtual bool      Load(int file_handle);
protected:
   virtual int       Compare(const CObject *node,int mode);
   //--- méthodes de création des instances de classe
   virtual CTreeNode* CreateSample();
  };
//+------------------------------------------------------------------+
//| Constructeur de la classe CMyTreeNode. |
//| ENTREE :  aucun. |
//| SORTIE : aucune. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
void CMyTreeNode::CMyTreeNode()
  {
//--- initialisation des données utilisateur
   m_long        =0;
   m_double      =0.0;
   m_string      ="";
   m_datetime    =0;
  }
//+------------------------------------------------------------------+
//| Comparaison avec un autre noeud par l'algorithme spécifié. |
//| ENTREE :  noeud - élément à comparer,                          |
//|         mode - identifiant de l'algorithme de comparaison. |
//| SORTIE : résultat de la comparaison (>0,0,<0). |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
int CMyTreeNode::Compare(const CObject *node,int mode)
  {
//--- le paramètre mode est ignoré, car l'algorithme de construction de l'arbre est le
   int res=0;
//--- cast explicite
   CMyTreeNode *n=node;
   res=(int)(m_long-n.m_long);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Création d'une nouvelle instance de classe. |
//| ENTREE :  aucun. |
//| SORTIE : pointeur sur la nouvelle instance de classe CMyTreeNode. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
CTreeNode* CMyTreeNode::CreateSample()
  {
   CMyTreeNode *result=new CMyTreeNode;
//---
   return(result);
```

© 2000-2015, MetaQuotes Software Corp.

```
  }
//+------------------------------------------------------------------+
//| Sauvegarde les données du noeud de l'arbre dans un fichier. |
//| ENTREE :  file_handle - handle d'un fichier ouvert en écriture. |
//| SORTIE : vrai si OK, sinon faux. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CMyTreeNode::Save(int file_handle)
  {
   uint i=0,len;
//--- vérifications préliminaires
   if(file_handle<0) return(false);
//--- écriture des données utilisateur
//--- écriture des variables personnelles de type long
   if(FileWriteLong(file_handle,m_long)!=sizeof(long))           return(false);
//--- écriture des variables personnelles de type double
   if(FileWriteDouble(file_handle,m_double)!=sizeof(double))     return(false);
//--- écriture des variables personnelles de type string
   len=StringLen(m_string);
//--- écriture de la longueur de la chaîne
   if(FileWriteInteger(file_handle,len,INT_VALUE)!=INT_VALUE)    return(false);
//--- écriture de la chaîne
   if(len!=0 && FileWriteString(file_handle,m_string,len)!=len) return(false);
//--- écriture des données personnelles de type datetime
   if(FileWriteLong(file_handle,m_datetime)!=sizeof(long))       return(false);
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Charge les données du noeud de l'arbre depuis un fichier. |
//| ENTREE :  file_handle - handle d'un fichier ouvert en lecture. |
//| SORTIE : vrai si OK, sinon faux. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CMyTreeNode::Load(int file_handle)
  {
   uint i=0,len;
//--- vérifications préliminaires
   if(file_handle<0) return(false);
//--- lecture
   if(FileIsEnding(file_handle)) return(false);
//--- lecture des variables personnelles de type char
//--- lecture des variables personnelles de type long
   m_long=FileReadLong(file_handle);
//--- lecture des variables personnelles de type double
   m_double=FileReadDouble(file_handle);
//--- lecture des variables personnelles de type string
//--- lecture de la longueur de la chaîne
   len=FileReadInteger(file_handle,INT_VALUE);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- lecture de la chaîne
   if(len!=0) m_string=FileReadString(file_handle,len);
   else       m_string="";
//--- lecture des variables personnelles de type datetime
   m_datetime=FileReadLong(file_handle);
//---
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Owner

Retourne le pointeur du noeud père.

```
CTreeNode*  Owner()
```

Valeur de Retour

Pointeur sur le noeud père.

Owner

Définit le pointeur du noeud père.

```
void  Owner(
   CTreeNode*  node      // noeud
   )
```

Paramètres

node

[in] Nouvelle valeur du pointeur du noeud père.

Valeur de Retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Left

Retourne le pointeur du noeud situé à gauche.

```
CTreeNode*  Left()
```

Valeur de Retour

Pointeur du noeud situé à gauche.

Left

Définit le pointeur du noeud situé à gauche.

```
void  Left(
   CTreeNode*  node      // noeud
   )
```

Paramètres

node

[in] Nouvelle valeur du pointeur du noeud de gauche.

Valeur de Retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Right

Retourne le pointeur du noeud situé à droite.

```
CTreeNode*  Right()
```

Valeur de Retour

Le pointeur sur le noeud situé à droite.

Right

Définit le pointeur du noeud situé à droite.

```
void  Right(
   CTreeNode*  node      // noeud
   )
```

Paramètres

node

[in] Nouvelle valeur du pointeur du noeud de droite.

Valeur de Retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Balance

Retourne la balance du noeud.

```
int  Balance() const
```

Valeur de Retour

Balance du noeud.

© 2000-2015, MetaQuotes Software Corp.

BalanceL

Retourne la balance de la sous-branche de gauche du noeud.

```
int  BalanceL() const
```

Valeur de Retour

Balance de la of the left sub-branch of the node.

© 2000-2015, MetaQuotes Software Corp.

BalanceR

Retourne la balance de la sous-branche de droite du noeud.

```
int  BalanceR() const
```

Valeur de Retour

Balance de la sous-branche de droite du noeud.

© 2000-2015, MetaQuotes Software Corp.

CreateSample

Crée un nouveau noeud modèle.

```
virtual CTreeNode*  CreateSample()
```

Valeur de Retour

Pointeur vers le nouveau noeud modèle ou NULL.

© 2000-2015, MetaQuotes Software Corp.

RefreshBalance

Recalcule la balance du noeud.

```
int  RefreshBalance()
```

Valeur de Retour

Balance du noeud.

© 2000-2015, MetaQuotes Software Corp.

GetNext

Retourne un pointeur vers le noeud suivant.

```
CTreeNode*  GetNext(
   CTreeNode*  node      // noeud
   )
```

Paramètres

node

[in] Noeud de départ de la recherche.

Valeur de Retour

Pointeur sur le noeud suivant.

© 2000-2015, MetaQuotes Software Corp.

SaveNode

// Sauvegarde les données du noeud dans un fichier.

```
bool  SaveNode(
   int  file_handle      // handle
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire ouvert précédemment en écriture.

Valeur de Retour

vrai en cas de succès, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

LoadNode

// Charge les données du noeud depuis un fichier.

```
bool  LoadNode(
   int          file_handle,      // handle
   CTreeNode*   main              // noeud
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire ouvert précédemment en lecture.

main

[in] Noeud à remplir.

Valeur de Retour

vrai en cas de succès, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type du noeud.

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type du noeud.

© 2000-2015, MetaQuotes Software Corp.

CTree

La classe CTree est un arbre binaire d'instances de classe CTreeNode et de ses descendants.

Description

La classe CTree permet de travailler avec un arbre binaire d'instances de la classe CTreeNode et de ses descendants. Les méthodes d'ajout, insertion et suppression des éléments et de recherche dans l'arbre sont implémentées dans cette classe. Elle implémente également les méthodes permettant de travailler avec les fichiers.

Noter que le mécanisme de gestion de la mémoire n'est pas implémenté dans la classe CTree (contrairement aux classes CList et CArrayObj). Tous les noeuds de l'arbre sont supprimés et la mémoire libérée.

Déclaration

```
class CTree : public CTreeNode
```

Titre

```
#include <Arrays\Tree.mqh>
```

Méthodes de Classe

Attributs	
Root	Retourne le noeud racine de l'arbre
Création d'u nouvel élément	
CreateElement	Création d'un nouveau noeud
Remplissage	
Insert	Ajout d'un noeud à un arbre
Suppression	
Detach	Détache de l'arbre un noeud spécifié
Delete	Supprime de l'arbre un noeud spécifié
Clear	Supprime tous les noeuds de l'arbre
Recherche	
Find	Recherche un noeud dans l'arbre à partir d'un modèle
Entrée/Sortie	
virtual Save	Sauvegarde les données de l'arbre dans un fichier
virtual Load	Charge les données d'un arbre depuis un fichier
virtual Type	Retourne l'identifiant du type de l'arbre

© 2000-2015, MetaQuotes Software Corp.

Les arbres de classes CTreeNode dérivant de CTree sont mis en application.

Les classes filles de la classe CTree doivent implémenter la méthode prédéfinie CreateElement qui crée un nouveau CTreeNode.

Considérons l'exemple d'une classe dérivée de la classe CTree.

```
//+------------------------------------------------------------------+
//|                                                      MyTree.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                              http://www.metaquotes.net/ |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\Tree.mqh>
#include "MyTreeNode.mqh"
//---
input int extCountedNodes = 100;
//+------------------------------------------------------------------+
//| Describe class CMyTree derived from CTree. |
//+------------------------------------------------------------------+
//| Class CMyTree.                                                   |
//| But : Construction et navigation d'une recherche binaire dans un arbre. |
//+------------------------------------------------------------------+
class CMyTree : public CTree
  {
public:
   //--- méthode de recherche dans l'arbre à partir de données spécifiques
   CMyTreeNode*        FindByLong(long find_long);
   //--- méthode de création de l'élément dans l'arbre
   virtual CTreeNode *CreateElement();
  };
//---
CMyTree MyTree;
//+------------------------------------------------------------------+
//| Création d'un nouveau noeud dans l'arbre. |
//| ENTREE :  aucun. |
//| SORTIE : pointeur vers le nouveau noeud de l'arbre en cas de succès, sinon NULL. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
CTreeNode *CMyTree::CreateElement()
  {
   CMyTreeNode *node=new CMyTreeNode;
//---
   return(node);
  }
//+------------------------------------------------------------------+
//| Cherche un élément à partir d'une valeur m_long. |
```

© 2000-2015, MetaQuotes Software Corp.

```
//| ENTREE :  find_long - valeur recherchée. |
//| SORTIE : pointeur sur l'élément trouvé, ou NULL. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
CMyTreeNode* CMyTree::FindByLong(long find_long)
  {
   CMyTreeNode *res=NULL;
   CMyTreeNode *node;
//--- crée un nouveau noeud pour passer le paramètre de recherche
   node=new CMyTreeNode;
   if(node==NULL) return(NULL);
   node.SetLong(find_long);
//---
   res=Find(node);
   delete node;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| script "test de la class CMyTree"                                |
//+------------------------------------------------------------------+
//---  tableau de chaînes d'initialisation
string str_array[11]={"p","oo","iii","uuuu","yyyyy","ttttt","rrrr","eee","ww","q","999
//---
int OnStart() export
  {
   int          i;
   uint         pos;
   int          beg_time,end_time;
   CMyTreeNode *node; //--- pointeur temporaire vers le modèle de la classe CMyTreeNod
//---
   printf("Start test %s.",__FILE__);
//--- Remplissage de MyTree avec les modèles de classe MyTreeNode.
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i++)
     {
      node=MyTree.CreateElement();
      if(node==NULL)
        {
         //--- sortie d'urgence
         printf("%s (%4d): create error",__FILE__,__LINE__);
         return(__LINE__);
        }
      NodeSetData(node,i);
      node.SetLong(i);
      MyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("Filling time of MyTree is %d ms.",end_time-beg_time);
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- Crée un arbre temporaire TmpMyTree.
   CMyTree TmpMyTree;
//--- Détache 50% des éléments de l'arbre (tous les noeuds ayant une valeur paire)
//--- et les ajoute à la table temporaire TmpMyTree.
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i+=2)
     {
      node=MyTree.FindByLong(i);
      if(node!=NULL)
         if(MyTree.Detach(node)) TmpMyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("Deletion time of %d elements from MyTree is %d ms.",extCountedNodes/2,end_t
//--- Retourne l'élément détaché
   node=TmpMyTree.Root();
   while(node!=NULL)
     {
      if(TmpMyTree.Detach(node)) MyTree.Insert(node);
      node=TmpMyTree.Root();
     }
//--- Appelle et vérifie la méthode Save(int file_handle);
   int file_handle;
   file_handle=FileOpen("MyTree.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!MyTree.Save(file_handle))
        {
         //--- erreur d'écriture dans le fichier
         //--- sortie d'urgence
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- fermeture du fichier avant de quitter !!!
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
     }
//--- Appelle et vérifie la méthode Load(int file_handle);
   file_handle=FileOpen("MyTree.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!TmpMyTree.Load(file_handle))
        {
         //--- erreur de lecture du fichier
         //--- sortie d'urgence
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- fermeture du fichier avant de quitter !!!
         FileClose(file_handle);
         return(__LINE__);
        }
```

© 2000-2015, MetaQuotes Software Corp.

```
      FileClose(file_handle);
     }
//---
   MyTree.Clear();
   TmpMyTree.Clear();
//---
   printf("End test %s. OK!",__FILE__);
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| Fonction de journalisation du contenu d'un noeud |
//+------------------------------------------------------------------+
void NodeToLog(CMyTreeNode *node)
  {
   printf("   %I64d,%f,'%s','%s'",
              node.GetLong(),node.GetDouble(),
              node.GetString(),TimeToString(node.GetDateTime()));
  }
//+------------------------------------------------------------------+
//| Fonction de "remplissage" d'un noeud avec des valeurs aléatoires |
//+------------------------------------------------------------------+
void NodeSetData(CMyTreeNode *node,int mode)
  {
   if(mode%2==0)
     {
      node.SetLong(mode*MathRand());
      node.SetDouble(MathPow(2.02,mode)*MathRand());
     }
   else
     {
      node.SetLong(mode*(long)(-1)*MathRand());
      node.SetDouble(-MathPow(2.02,mode)*MathRand());
     }
   node.SetString(str_array[mode%10]);
   node.SetDateTime(10000*mode);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Root

Retourne le noeud racine de l'arbre.

```
CTreeNode*  Root() const
```

Valeur de Retour

Pointeur vers le noeud racine de l'arbre.

© 2000-2015, MetaQuotes Software Corp.

CreateElement

Crée un nouvelle instance de noeud.

```
virtual CTreeNode*  CreateElement()
```

Valeur de Retour

Pointeur vers la nouvelle instance du noeud ou NULL.

© 2000-2015, MetaQuotes Software Corp.

Insert

Ajoute un noeud à l'arbre.

```
CTreeNode*  Insert(
   CTreeNode*  new_node      // noeud
   )
```

Paramètres

new_node

[in] pointeur vers le noeud à insérer dans l'arbre.

Valeur de Retour

Pointeur sur le noeud père ou NULL.

© 2000-2015, MetaQuotes Software Corp.

Detach

Détache un noeud spécifié de l'arbre.

```
bool  Detach(
   CTreeNode*  node      // noeud
   )
```

Paramètres

node

[in] Pointeur vers le noeud à détacher.

Valeur de Retour

vrai en cas de succès, faux sinon.

Note

Après avoir détaché le noeud, le pointeur n'est pas libéré. L'arbre est équilibré.

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime de l'arbre un noeud spécifié.

```
bool  Delete(
   CTreeNode*  node      // noeud
   )
```

Paramètres

node

[in] Pointeur vers le noeud à supprimer.

Valeur de Retour

vrai en cas de succès, faux sinon.

Note

Après la suppression du noeud, le pointeur est libéré. L'arbre est équilibré.

© 2000-2015, MetaQuotes Software Corp.

Clear

Supprime tous les noeuds de l'arbre.

```
void  Clear()
```

Valeur de Retour

Aucune.

Note

Après la suppression des noeuds, tous les pointeurs sont libérés.

© 2000-2015, MetaQuotes Software Corp.

Find

Recherche un noeud dans l'arbre à partir d'un modèle.

```
CTreeNode*  Find(
   CTreeNode*  node      // noeud
   )
```

Paramètres

node

[in] Noeud contenant les données à chercher.

Valeur de Retour

Pointeurr sur le noeud s'il a été trouvé, NULL sinon.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les données de l'arbre dans un fichier.

```
virtual bool  Save(
   int  file_handle      // handle
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire ouvert précédemment en écriture.

Valeur de Retour

vrai en cas de succès, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les données d'un arbre depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // handle
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire ouvert précédemment en lecture.

Valeur de Retour

vrai en cas de succès, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'arbre.

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type de l'arbre.

© 2000-2015, MetaQuotes Software Corp.

Graphic Objects

This section contains the technical details of working with classes of graphical objects and a description of the relevant components of the standard library MQL5.

The use of classes of graphical objects, will save time when creating custom programs (scripts, expert).

Standard library MQL5 (in terms of graphical objects) is placed in the working directory of the terminal in the folder Include \ ChartObjects.

Class/Group	Description
Base class for graphical object CChartObject	Base class of a graphic object
Lines	Group classes "Lines"
Channels	Group classes "Channels"
Gann Tools	Group classes "Gann"
Fibonacci Tools	Group classes "Fibonacci"
Elliott Tools	Group classes "Elliott"
Shapes	Group classes "Shapes"
Arrows	Group classes "Arrows"
Controls	Group classes "Controls"

© 2000-2015, MetaQuotes Software Corp.

CChartObject

La classe CChartObject est la classe de base des objets graphique du type de graphique de la bibliothèque MQL5 Standard.

Description

La classe CChartObject fournit un accès simplifié aux fonctions de l'API MQL5 à tous ses descendants.

Déclaration

```
class CChartObject : public CObject
```

Titre

```
#include <ChartObjects\ChartObject.mqh>
```

Méthodes de Classe

Attributs	
ChartId	Retourne l'identifiant du graphique contenant un graphique
Window	Retourne le nombre de fenêtres dans lesquelles l'objet graphique est un graphique
Name	Retourne/définit le nom de l'objet graphique
NumPoints	Retourne le nombre de points d'ancrage
Assign	
Attach	Assigne un graphique
SetPoint	Définit le point d'ancrage
Delete	
Delete	Supprime un graphique
Detach	Détache un graphique
Shift	
ShiftObject	Déplacement relatif d'un objet
ShiftPoint	Déplacement relatif du point de l'objet
Propriétés de l'objet	
Time	Retourne/Définit les coordonnées date/heure du point de l'objet
Price	Retourne/Définit les coordonnées prix du point de l'objet
Color	Retourne/Définit la couleur de l'objet

© 2000-2015, MetaQuotes Software Corp.

Style	Retourne/Définit le style de ligne de l'objet
Width	Retourne/Définit la largeur de ligne de l'objet
BackGround	Retourne/Définit le flag pour l'affichage de l'arrière plan d'un objet
Selected	Retourne/Définit le flag "selected" de l'objet graphique.
Selectable	Retourne/Définit le flag autorisant ou pas la sélection de l'objet
Description	Retourne/Définit le texte de l'objet
Tooltip	Retourne/Définit l'infobulle de l'objet
Timeframes	Retourne/Définit le masque de visibilité des flags de l'objet
Z_Order	Retourne/Définit la priorité du clic dans un graphique
CreateTime	Retourne l'heure de création d'un objet
Propriétés des niveaux de l'objet	
LevelsCount	Retourne/Définit le nombre de niveaux d'un objet
LevelColor	Retourne/Définit la couleur de ligne du niveau
LevelStyle	Retourne/Définit le style de ligne du niveau
LevelWidth	Retourne/Définit la largeur de ligne du niveau
LevelValue	Retourne/Définit la valeur du niveau
LevelDescription	Retourne/Définit le texte du niveau
Accès aux fonctions de l'API MQL5	
GetInteger	Retourne la valeur des propriétés de l'objet
SetInteger	Définit les propriétés de l'objet
GetDouble	Retourne la valeur des propriétés de l'objet
SetDouble	Définit les propriétés de l'objet
GetString	Retourne la valeur des propriétés de l'objet
SetString	Définit les propriétés de l'objet
Entrée/Sortie	
virtual Save	Méthode virtuelle de sauvegarde d'un fichier
virtual Load	Méthode virtuelle de lecture d'un fichier
virtual Type	Méthode virtuelle d'identification

© 2000-2015, MetaQuotes Software Corp.

Classes dérivées :

- CChartObjectArrow
- CChartObjectBitmap
- CChartObjectBmpLabel
- CChartObjectCycles
- CChartObjectElliottWave3
- CChartObjectEllipse
- CChartObjectFiboArc
- CChartObjectFiboFan
- CChartObjectFiboTimes
- CChartObjectHLine
- CChartObjectRectangle
- CChartObjectSubChart
- CChartObjectText
- CChartObjectTrend
- CChartObjectTriangle
- CChartObjectVLine

© 2000-2015, MetaQuotes Software Corp.

ChartId

Retourne l'identifiant du graphique auquel l'objet graphique appartient.

```
long  ChartId() const
```

Valeur de Retour

Identifiant du Graphique dans lequel se trouve l'objet graphique. Si aucun objet n'est trouvé, retourne -1.

Exemple :

```
//--- exemple d'utilisation de CChartObject::ChartId
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- recupère l'identifiant du Graphique de l'objet
   long chart_id=object.ChartId();
  }
```

© 2000-2015, MetaQuotes Software Corp.

Window

Retourne le nombre de fenêtres dans lesquelles l'objet graphique est un graphique

```
int  Window() const
```

Valeur de Retour

Numéro de la fenêtre du graphique où l'objet graphique est positionné (0 - fenêtre principale). Si aucun objet n'est trouvé, retourne -1.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Window
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère la fenêtre de l'objet graphique
   int window=object.Window();
  }
```

© 2000-2015, MetaQuotes Software Corp.

Name (Méthode "Get")

Retourne le nom de l'objet graphique.

```
string  Name() const
```

Valeur de Retour

Nom de l'objet graphique lié à l'instance de classe. Si l'objet n'est pas trouvé, retourne NULL.

Name (Méthode "Set")

Définits le nom de l'objet graphique.

```
bool  Name(
   string  name      // nouveau nom
   )
```

Paramètres

name

[in] Le nouveau nom de l'objet graphique.

Valeur de Retour

vrai si réalisé avec succès, faux si le nom ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Name
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- retourne le nom de l'objet graphique
   string object_name=object.Name();
   if(object_name!="MyChartObject")
     {
     //--- définit le nom de l'objet graphique
     object.Name("MyChartObject");
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

NumPoints

Retourne le nombre de points d'ancrage de l'objet graphique.

```
int  NumPoints() const
```

Valeur de Retour

Nombre de points liant un objet graphique lié à une instance de classe. Si aucun objet n'est assigné, retourne 0.

Exemple :

```
//--- exemple d'utilisation de CChartObject::NumPoints
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le nombre de points de l'objet graphique
   int points=object.NumPoints();
  }
```

© 2000-2015, MetaQuotes Software Corp.

Attach

Assigne un object graphique à une instance de la classe.

```
bool  Attach(
   long     chart_id,      // Identifiant du graphique
   string   name,          // Nom de l'objet
   int      window,        // Fenêtre du graphique
   int      points         // Nombre de points
   )
```

Paramètres

chart_id

[out] Identifiant du graphique.

name

[in] Nom de l'objet graphique.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

points

[in] Nombre de points d'ancrage de l'objet graphique.

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Attach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- assigne l'objet Graphique
   if(!object.Attach(ChartID(),"MyObject",0,2))
     {
      printf("Erreur d'assignation de l'objet");
      return;
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

SetPoint

Définit les nouvelles coordonnées du point d'ancrage de l'objet graphique.

```
bool  SetPoint(
   int       point,          // Numéro du point
   datetime  new_time,       // Coordoonées en date/heure
   double    new_price       // Coordonnées en prix
   )
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

new_time

[in] Nouvelle valeur de la coordonnée date/heure du point d'ancrage spécifié.

new_price

[in] Nouvelle valeur de la coordonnée prix du point d'ancrage spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si les coordonnées du point ne peuvent pas être modifiées.

Exemple :

```
//--- exemple d'utilisation de CChartObject::SetPoint
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   if(object.NumPoints()>0)
     {
      //--- définit le point de l'objet
      object.SetPoint(0,CurrTime(),price);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime un objet graphique attaché au graphique.

```
bool  Delete()
```

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Delete
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- détache l'objet graphique
   if(!object.Delete())
     {
      printf("Erreur de suppression de l'objet");
      return;
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Detach

Détache l'objet graphique.

```
void  Detach()
```

Valeur de Retour

Aucune.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Detach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- détache l'objet graphique
   object.Detach();
  }
```

© 2000-2015, MetaQuotes Software Corp.

ShiftObject

Déplace un objet graphique.

```
bool  ShiftObject(
   datetime  d_time,      // Incrément des coordonnées date/heure
   double    d_price      // Incrément des coordonnées prix
   )
```

Paramètres

d_time

[in] Incrément de la coordonnée date/heure de tous les points d'ancrage.

d_price

[in] Incrément de la coordonnée prix de tous les points d'ancrage.

Valeur de Retour

vrai si réalisé avec succès, faux si l'objet ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CChartObject::ShiftObject
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //--- déplace l'objet graphique
   object.ShiftObject(d_time,d_price);
  }
```

© 2000-2015, MetaQuotes Software Corp.

ShiftPoint

Décale un point d'ancrage spécifié de l'objet graphique.

```
bool  ShiftPoint(
   int       point,       // Numéro du point
   datetime  d_time,      // Incrément des coordonnées date/heure
   double    d_price      // Incrément des coordonnées prix
   )
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

d_time

[in] Incrément de la coordonnée date/heure du point spécifié.

d_price

[in] Incrément de la coordonnée prix du point spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si le point ne peut pas être déplacé.

Exemple :

```
//--- exemple d'utilisation de CChartObject::ShiftPoint
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //---
   if(object.NumPoints()>0)
     {
      //--- décale le point de l'objet graphique
      object.ShiftPoint(0,d_time,d_price);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Time (Méthode "Get")

Retourne la coordonnée date/heure du point d'ancrage spécifié d'un objet graphique.

```
datetime  Time(
   int  point      // Numéro du point
   ) const
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

Valeur de Retour

Coordonnées date/heure du point d'ancrage spécifié de l'objet graphique lié à une instance de classe. Si aucun objet n'est assigné ou si aucun point ne correspond au numéro spécifié, retourne 0.

Time (Méthode "Set")

Retourne la coordonnée date/heure du point d'ancrage spécifié d'un objet graphique.

```
bool  Time(
   int       point,        // Numéro du point
   datetime  new_time      // Date/heure
   )
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

new_time

[in] Nouvelle valeur de la coordonnée date/heure du point d'ancrage spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si la coordonnée ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Time
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- récupère l'heure du point de l'objet
      datetime point_time=object.Time(i);
      if(point_time==0)
```

© 2000-2015, MetaQuotes Software Corp.

```
         {
          //--- définit la date/heure du point de l'objet
          object.Time(i,TimeCurrent());
         }
      }
   }
```

© 2000-2015, MetaQuotes Software Corp.

Price (Méthode "Get")

Retourne la coordonnée prix du point d'ancrage spécifié d'un objet graphique.

```
double  Price(
   int  point      // Numéro du point
   ) const
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

Valeur de Retour

Coordonnées prix du point d'ancrage spécifié de l'objet graphique lié à une instance de classe. Si aucun objet n'est assigné ou si aucun point ne correspond au numéro spécifié, retourne EMPTY_VALUE.

Price (Méthode "Set")

Définit les coordonnées prix du point d'ancrage spécifié de l'objet graphique.

```
bool  Price(
   int     point,          // Numéro du point
   double  new_price       // Prix
   )
```

Paramètres

point

[in] Numéro d'un point d'ancrage.

new_price

[in] Nouvelle valeur de la coordonnée prix du point d'ancrage spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si la coordonnée ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Price
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- récupère le prix du point de l'objet
```

© 2000-2015, MetaQuotes Software Corp.

```
      double point_price=object.Price(i);
      if(point_price!=price)
        {
         //--- définit le prix du point de l'objet graphique
         object.Price(i,price);
        }
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Color (Méthode "Get")

Retourne la couleur de ligne d'un objet graphique.

```
color  Color() const
```

Valeur de Retour

Couleur de ligne de l'objet graphique assigné à l'instance de classe. Si aucun objet n'est assigné, retourne CLR_NONE.

Color (Méthode "Set")

Définit la couleur de ligne de l'objet graphique.

```
bool  Color(
   color  new_color      // Nouvelle couleur
   )
```

Paramètres

new_color

[in] Nouvelle couleur de ligne.

Valeur de Retour

vrai si réalisé avec succès, faux si la couleur ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Color
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--récupère la couleur de l'objet graphique
   color object_color=object.Color();
   if(object_color!=clrRed)
     {
     //--- définit la couleur de l'objet graphique
     object.Color(clrRed);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Style (Méthode "Get")

Retourne le style de ligne d'un objet graphique.

```
ENUM_LINE_STYLE  Style() const
```

Valeur de Retour

Style de ligne de l'objet graphique assigné à l'instance de classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Style (Méthode "Set")

Définit le style de ligne de l'objet graphique.

```
bool  Style(
   ENUM_LINE_STYLE  new_style      // Style
   )
```

Paramètres

new_style

[in] Nouvelle valeur du style de ligne.

Valeur de Retour

vrai si réalisé avec succès, faux si le style ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Style
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le style de ligne de l'objet
   ENUM_LINE_STYLE style=object.Style();
   if(style!=STYLE_SOLID)
     {
      //--- définit le style de ligne de l'objet graphique
      object.Style(STYLE_SOLID);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Width (Méthode "Get")

Retourne la largeur de ligne d'un objet graphique.

```
int  Width() const
```

Valeur de Retour

La largeur de ligne d'un objet graphique attaché à une instance de la classe. Si aucun objet n'est assigné, retourne -1.

Width (Méthode "Set")

Définit la largeur de ligne d'un objet graphique.

```
bool  Width(
   int  new_width      // Epaisseur
   )
```

Paramètres

new_width

[in] Nouvelle valeur d'épaisseur de ligne.

Valeur de Retour

vrai si réalisé avec succès, faux si l'épaisseur de ligne ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Width
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère la largeur de l'objet
   int width=object.Width();
   if(width!=1)
     {
      //--- définit la largeur de l'objet
      object.Width(1);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Background (Méthode "Get")

Retourne le flag pour dessiner un objet graphique en arrière plan.

```
bool  Background() const
```

Valeur de Retour

Flags de dessin d'un objet graphique attaché à une instance de la classe en arrière plan. Si aucun objet n'est attaché, retourne faux.</t3>

Background (Méthode "Set")

Définit le flag pour le dessin d'un objet graphique en arrière plan.

```
bool  Background(
   bool  background      // Valeur du flag
   )
```

Paramètres

background

[in] Nouvelle valeur du flag de dessin de l'objet graphique en arrière plan.

Valeur de Retour

vrai si réalisé avec succès, faux si le flag ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Background
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- recupère le flag d'affichage en arrière plan de l'objet graphique
   bool background_flag=object.Background();
   if(!background_flag)
     {
     //--- définit le flag d'affichage en arrière plan de l'objet graphique
     object.Background(true);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Selected (Méthode "Get")

Retourne le flag indiquant qu'un objet graphique est sélectionné. En d'autres termes, si l'objet graphique est sélectionné ou pas.

```
bool  Selected() const
```

Valeur de Retour

L'état que l'objet, attaché à une instance de la classe, est sélectionné. Si aucun objet n'est assigné, retourne faux.

Selected (Méthode "Set")

Définit le flag indiquant qu'un objet graphique est sélectionné.

```
bool  Selected(
   bool  selected      // Valeur du flag
   )
```

Paramètres

selected

[in] Nouvelle valeur du flag.

Valeur de Retour

vrai si réalisé avec succès, faux si le flag ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Selected
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le flag "selected" de l'objet graphique
   bool selected_flag=object.Selected();
   if(selected_flag)
     {
     //--- définit le flag "selected" de l'objet graphique
     object.Selected(false);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Selectable (Méthode "Get")

Retourne le flag indiquant la possibilité d'un objet graphique à être sélectionné. En d'autres termes, si l'objet graphique peut être sélectionné ou pas.

```
bool  Selectable() const
```

Valeur de Retour

Flag indiquant la possibilité de l'objet, attaché à une instance de classe, à être sélectionné. Si aucun objet n'est assigné, retourne faux.

Selectable (Méthode "Set")

Définit le flag indiquant la possibilité d'un objet graphique à être sélectionné.

```
bool  Selectable(
   bool  selectable      // Valeur du flag
   )
```

Paramètres

selectable

[in] Nouvelle valeur du flag.

Valeur de Retour

vrai si réalisé avec succès, faux si le flag ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Selectable
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le flag "selectable" de l'objet graphique
   bool selectable_flag=object.Selectable();
   if(selectable_flag)
     {
     //--- définit le flag "selectable" de l'objet graphique
     object.Selectable(false);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Description (Méthode "Get")

Retourne une description (texte) d'un objet graphique.

```
string  Description() const
```

Valeur de Retour

Description (texte) de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné, retourne NULL.

Description (Méthode "Set")

Définit la description (texte) d'un objet graphique.

```
bool  Description(
   string  text      // Texte
   )
```

Paramètres

text

[in] Nouvelle description (texte).

Valeur de Retour

vrai si réalisé avec succès, faux si la description ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Description
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère la description de l'objet graphique
   string description=object.Description();
   if(description=="")
     {
      //--- définit la description de l'objet
      object.Description("MyObject");
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Tooltip (Méthode "Get")

Retourne le texte de l'infobulle d'un objet graphique.

```
string  Tooltip() const
```

Valeur de retour

Le texte de l'infobulle d'un objet graphique attaché à une instance de la classe. Si aucun objet n'est assigné, retourne NULL.

Tooltip (Méthode "Set")

Définit le texte de l'infobulle d'un objet graphique.

```
bool  Tooltip(
   string  new_tooltip      // nouveau texte de l'infobulle
   )
```

Paramètres

new_tooltip

[in] Nouveau texte d'une infobulle.

Valeur de retour

vrai si réalisé avec succès, faux si l'infobulle n'a pas pu être modifiée.

Note :

Si la propriété n'est pas définie, alors l'infobulle générée automatiquement par le terminal est affichée. L'infobulle peut être désactivée en utilisant la valeur "\n" (retour à la ligne).

© 2000-2015, MetaQuotes Software Corp.

Timeframes (Méthode "Get")

Retourne les flags de visibilité d'un objet graphique.

```
int  Timeframes() const
```

Valeur de Retour

Flags de visibilité d'un objet graphique attaché à une instance de la classe. Si aucun objet n'est assigné, retourne 0.

Timeframes (Méthode "Set")

Définit les flags de visibilité d'un objet graphique.

```
bool  Timeframes(
   int  new_timeframes      // Flags de visibilité
   )
```

Paramètres

new_timeframes

[in] Nouveau flags de visibilité de l'objet graphique.

Valeur de Retour

vrai si réalisé avec succès, faux si les flags de visibilité ne peuvent pas être changés.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Timeframes
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère les espaces de temps de l'objet graphique
   int timeframes=object.Timeframes();
   if(!(timeframes&OBJ_PERIOD_H1))
     {
      //---définit les espaces de temps de l'objet graphique
      object.Timeframes(timeframes|OBJ_PERIOD_H1);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Z_Order (Méthode "Get")

Retourne la priorité de l'objet graphique lors d'un clic sur le graphique (CHARTEVENT_CLICK).

```
long  Z_Order() const
```

Valeur de Retour

Priorité d'un objet graphique assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

Z_Order (Méthode "Set")

Définit la priorité d'un objet graphique lors d'un clic sur le graphique (CHARTEVENT_CLICK).

```
bool  Z_Order(
   long  value      // nouvelle priorité
   )
```

Paramètres

value

[in] Nouvelle priorité d'un objet graphique lors d'un clic sur le graphique (CHARTEVENT_CLICK).

Valeur de Retour

vrai si réalisé avec succès, faux si la priorité ne peut pas être modifiée.

Note

Z_Order est la priorité d'un objet graphique à recevoir les évènements de clic sur un graphique (CHARTEVENT_CLICK). En définissant une valeur supérieure à 0 (valeur par défaut), vous pouvez augmenter la priorité de l'objet.

© 2000-2015, MetaQuotes Software Corp.

CreateTime

Retourne la date/heure de création d'un objet graphique.

```
datetime  CreateTime() const
```

Valeur de Retour

Heure de création de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

Exemple :

```
//--- exemple d'utilisation de CChartObject::CreateTime
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère l'heure de création de l'objet graphique
   datetime create_time=object.CreateTime();
  }
```

© 2000-2015, MetaQuotes Software Corp.

LevelsCount (Méthode "Get")

Retourne le nombre de niveaux de l'objet graphique.

```
int  LevelsCount() const
```

Valeur de Retour

Nombre de niveaux de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

LevelsCount (Méthode "Set")

Définit le nombre de niveaux de l'objet graphique.

```
bool  LevelsCount(
   int  levels      // Nombre de niveaux
   )
```

Paramètres

levels

[in] Le nouveau nombre de niveaux de l'objet graphique.

Valeur de Retour

vrai si réalisé avec succès, faux si le nombre de niveaux ne peut pas être changé.

Exemple :

```
//--- exemple d'utilisation de CChartObject::LevelsCount
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le nombre de niveaux du graphique
   int levels_count=object.LevelsCount();
   //--- ajoute un niveau
   object.LevelsCount(levels_count+1);
  }
```

© 2000-2015, MetaQuotes Software Corp.

LevelColor (Méthode "Get")

Retourne la couleur de ligne du niveau spécifié de l'objet graphique.

```
color  LevelColor(
   int  level      // Numéro de niveau
   ) const
```

Paramètres

level

[in] Numéro du niveau.

Valeur de Retour

Couleur de ligne du niveau spécifié de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné ou si aucun niveau ne correspond au numéro spécifié, retourne CLR_NONE.

LevelColor (Méthode "Set")

Définit la couleur de ligne du niveau spécifié de l'objet graphique.

```
bool  LevelColor(
   int    level,          // Numéro de niveau
   color  new_color       // Nouvelle couleur
   )
```

Paramètres

level

[in] Numéro du niveau.

new_color

[in] Nouvelle couleur de ligne du niveau spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si la couleur ne peut pas être modifiée.

Exemple :

```
//--- exemple for CChartObject::LevelColor
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère la couleur du niveau de l'objet graphique
      color level_color=object.LevelColor(i);
      if(level_color!=clrRed)
```

© 2000-2015, MetaQuotes Software Corp.

```
         {
          //--- modifie la couleur du niveau de l'objet graphique
          object.LevelColor(i,clrRed);
         }
      }
   }
```

© 2000-2015, MetaQuotes Software Corp.

LevelStyle (Méthode "Get")

Retourne le style de ligne du niveau spécifié de l'objet graphique.

```
ENUM_LINE_STYLE  LevelStyle(
   int  level      // Numéro de niveau
   ) const
```

Paramètres

level

[in] Numéro du niveau.

Valeur de Retour

Style de ligne du niveau spécifié de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné ou si aucun niveau ne correspond au numéro spécifié, retourne WRONG_VALUE.

LevelStyle (Méthode "Set")

Définit le style de ligne du niveau spécifié de l'objet graphique.

```
int  LevelStyle(
   int              level,     // Numéro du niveau
   ENUM_LINE_STYLE  style      // Style de ligne
   )
```

Paramètres

level

[in] Numéro du niveau.

style

[in] Nouveau style de ligne du niveau spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si le style ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::LevelStyle
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère le style niveau de l'objet graphique
      ENUM_LINE_STYLE level_style=object.LevelStyle(i);
      if(level_style!=STYLE_SOLID)
```

© 2000-2015, MetaQuotes Software Corp.

```
         {
          //--- définit le style du niveau de l'objet graphique
          object.LevelStyle(i,STYLE_SOLID);
         }
      }
   }
```

© 2000-2015, MetaQuotes Software Corp.

LevelWidth (Méthode "Get")

Retourne l'épaisseur de ligne du niveau spécifié de l'objet graphique.

```
int  LevelWidth(
   int  level      // Numéro de niveau
   ) const
```

Paramètres

level

[in] Numéro du niveau.

Valeur de Retour

Epaisseur de ligne du niveau spécifié de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné ou si aucun niveau ne correspond au numéro spécifié, retourne -1.

LevelWidth (Méthode "Set")

Définit l'épaisseur de la ligne du niveau spécifié de l'objet graphique.

```
bool  LevelWidth(
   int  level,          // Numéro de niveau
   int  new_width       // Nouvelle largeur
   )
```

Paramètres

level

[in] Numéro du niveau.

new_width

[in] Nouvelle épaisseur de ligne du niveau spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si l'épaisseur de ligne ne peut pas être modifié.

Exemple :

```
//--- exemple d'utilisation de CChartObject::LevelWidth
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère l'épaisseur du niveau de l'objet graphique
      int level_width=object.LevelWidth(i);
      if(level_width!=1)
```

© 2000-2015, MetaQuotes Software Corp.

```
         {
          //--- définit l'épaisseur du niveau de l'objet graphique
          object.LevelWidth(i,1);
         }
      }
   }
```

© 2000-2015, MetaQuotes Software Corp.

LevelValue (Méthode "Get")

Retourne la valeur du niveau spécifié de l'objet graphique.

```
double  LevelValue(
   int  level      // Numéro de niveau
   ) const
```

Paramètres

level

[in] Numéro du niveau.

Valeur de Retour

La valeur du niveau de l'objet graphique lié à l'instance de classe. Si aucun objet n'est assigné ou si aucun niveau ne correspond au numéro spécifié, retourne EMPTY_VALUE.

LevelValue (Méthode "Set")

Définit la valeur du niveau de l'objet graphique.

```
bool  LevelValue(
   int     level,          // Numéro du niveau
   double  new_value       // Nouvelle valeur
   )
```

Paramètres

level

[in] Numéro du niveau.

new_value

[in] Nouvelle valeur du niveau spécifié.

Valeur de Retour

vrai si réalisé avec succès, faux si la valeur ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::LevelValue
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère la valeur du niveau de l'objet graphique
      double level_value=object.LevelValue(i);
      if(level_value!=0.1*i)
```

© 2000-2015, MetaQuotes Software Corp.

```
         {
          //--- définit la valeur du niveau de l'objet graphique
          object.LevelValue(i,0.1*i);
         }
      }
   }
```

© 2000-2015, MetaQuotes Software Corp.

LevelDescription (Méthode "Get")

Retourne la description du niveau de l'objet graphique.

```
string  LevelDescription(
   int  level       // Numéro du niveau
   ) const
```

Paramètres

level

[in] Numéro du niveau de l'objet graphique.

Valeur de Retour

Description du niveau de l'objet graphique lié à l'instance de class. Si aucun objet n'est assigné ou si aucun niveau ne correspond au numéro spécifié, retourne NULL.

LevelDescription (Méthode "Set")

Définit la description du niveau de l'objet graphique.

```
bool  LevelDescription(
   int     level ,     // Numéro du niveau
   string  text        // Texte
   )
```

Paramètres

level

[in] Numéro du niveau de l'objet graphique.

text

[in] Nouvelle description du niveau de l'objet graphique.

Valeur de Retour

vrai si réalisé avec succès, faux si la description ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::LevelDescription
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère la descrription du niveau de l'objet graphique
      string level_description=object.LevelDescription(i);
      if(level_description=="")
```

© 2000-2015, MetaQuotes Software Corp.

```
         {
          //--- modifie la description du niveau de l'objet graphique
          object.LevelDescription(i,"Level_"+IntegerToString(i));
         }
      }
   }
```

© 2000-2015, MetaQuotes Software Corp.

GetInteger

Fournit un accès simplifié à la fonction ObjectGetInteger() de l'API MQL5 pour les propriétés de type integer (de type bool, char, uchar, short, ushort, int, uint, ling, ulong, datetime, color) liées à une instance d'un objet graphique. Deux signatures existent :

Récupérer la valeur de la propriété sans en vérifier l'exactitude

```
long  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,          // Identifiant de la propriété de ty
   int                           modifier=-1       // Modificateur
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer.

modifier=-1

[in] Modificateur (index) de la propriété de type integer.

Valeur de Retour

En cas de succès, retourne la valeur de la propriété de type integer ; en cas d'erreur, retourne 0.

Récupérer la valeur de la propriété en vérifiant son exactitude

```
bool  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,       // Identifiant de la propriété de type
   int                           modifier,      // Modificateur
   long&                         value          // Référence à la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer de l'objet.

modifier

[in] Modificateur (index) de la propriété de type integer.

value

[out] Référence à une variable permettant de récupérer la valeur de la propriété.

Valeur de Retour

vrai si réalisé avec succès, faux si la propriété de type integer ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::GetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

© 2000-2015, MetaQuotes Software Corp.

```
{
 CChartObject object;
 //--- récupère la couleur de l'objet par la méthode simple
 printf("Objects color is %s",ColorToString(object.GetInteger(OBJPROP_COLOR),true));
 //--- récupère la couleur de l'objet par la méthode classique
 long color_value;
 if(!object.GetInteger(OBJPROP_COLOR,0,color_value))
   {
    printf("Get integer property error %d",GetLastError());
    return;
   }
 else
    printf("Objects color is %s",color_value);
 for(int i=0;i<object.LevelsCount();i++)
   {
    //--- récupère les largeurs des lignes de niveaux par la méthode simple
    printf("Level %d width is %d",i,object.GetInteger(OBJPROP_LEVELWIDTH,i));
    //--- récupère les largeurs des lignes de niveaux par la méthode classique
    long width_value;
    if(!object.GetInteger(OBJPROP_LEVELWIDTH,i,width_value))
      {
       printf("Get integer property error %d",GetLastError());
       return;
      }
    else
       printf("Level %d width is %d",i,width_value);
   }
}
```

© 2000-2015, MetaQuotes Software Corp.

SetInteger

Fournit un accès simplifié aux fonctions de l'API MQL5 ObjectSetInteger() pour les propriétés de type integer (de type bool, char, uchar, short, ushort, int, uint, ling, ulong, datetime, color) liées à une instance d'un objet graphique. Cette fonction peut être appelée de 2 façons :

Définition de la valeur d'une propriété ne nécessitant pas de modificateur

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // Identifiant de la propriété de type
   long                           value         // Valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer de l'objet.

value

[in] Nouvelle valeur pour la propriété de type integer et mutable.

Définition de la valeur d'une propriété en indiquant un modificateur

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // Identifiant de la propriété de type
   int                            modifier,     // Modificateur
   long                           value         // Valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer de l'objet.

modifier

[in] Modificateur (index) de la propriété de type integer.

value

[in] Nouvelle valeur pour la propriété de type integer et mutable.

Valeur de Retour

vrai si réalisé avec succès, faux si la propriété de type integer ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::SetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- définit la nouvelle couleur de l'objet graphique
```

© 2000-2015, MetaQuotes Software Corp.

```
   if(!object.SetInteger(OBJPROP_COLOR,clrRed))
     {
      printf("Set integer property error %d",GetLastError());
      return;
     }
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- définit la largeur des lignes de niveau
      if(!object.SetInteger(OBJPROP_LEVELWIDTH,i,i))
        {
         printf("Set integer property error %d",GetLastError());
         return;
        }
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

GetDouble

Fournit un accès simplifié aux fonctions de l'API MQL5 ObjectGetDouble() pour récupérer les valeurs de type double (float ou double) de l'objet graphique assigné à l'instance de classe. Deux signatures existent :

Récupérer la valeur de la propriété sans en vérifier l'exactitude

```
double  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,          // Identifiant de la propriété de typ
   int                          modifier=-1      // Modificateur
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double.

modifier=-1

[in] Modificateur (index) de la propriété de type double.

Valeur de Retour

En cas de succès, retourne la valeur de la propriété de type double ; en cas d'erreur, retourne EMPTY_VALUE.

Récupérer la valeur de la propriété en vérifiant son exactitude

```
bool  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,      // Identifiant de la propriété de type d
   int                          modifier,     // Modificateur
   double&                      value         // Référence vers la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double.

modifier

[in] Modificateur (index) de la propriété de type double.

value

[out] Référence à une variable permettant de récupérer la valeur de la propriété.

Valeur de Retour

vrai si réalisé avec succès, faux si l'objet ne peut pas renvoyer une valeur de type double.

Exemple :

```
//--- exemple d'utilisation de CChartObject::GetDouble
#include <ChartObjects\ChartObject.mqh>
//---
```

© 2000-2015, MetaQuotes Software Corp.

```
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère les valeurs de niveaux par la méthode simple
      printf("Level %d value=%f",i,object.GetDouble(OBJPROP_LEVELVALUE,i));
      //--- get levels value by classic method
      double value;
      if(!object.SetDouble(OBJPROP_LEVELVALUE,i,value))
        {
         printf("Get double property error %d",GetLastError());
         return;
        }
      else
         printf("Level %d value=%f",i,value);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

SetDouble

Fournit un accès simplifié aux fonctions de l'API MQL5 ObjectSetDouble() pour changer les propriétés de type double (type float ou double) lié à une instance de la classe de l'objet graphique. Deux signatures existent :

Définition de la valeur d'une propriété ne nécessitant pas de modificateur

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,      // Identifiant de propriété de type doub
   double                        value         // Valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double.

value

[in] Nouvelle valeur des propriétés de type double et mutables.

Définition de la valeur d'une propriété en indiquant un modificateur

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,       // Identifiant de la propriété de type
   int                           modifier,      // Modificateur
   double                        value          // Valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double.

modifier

[in] Modificateur (index) de la propriété de type double.

value

[in] Nouvelle valeur de la propriété de type double et mutable.

Valeur de Retour

vrai si réalisé avec succès, faux si la propriété de type double ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::SetDouble
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
//---
```

© 2000-2015, MetaQuotes Software Corp.

```
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- définit la valeur du niveau de l'objet graphique
      if(!object.SetDouble(OBJPROP_LEVELVALUE,i,0.1*i))
        {
         printf("Set double property error %d",GetLastError());
         return;
        }
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

GetString

Fournit un accès simplifié aux fonctions de l'API MQL5 ObjectGetString() pour récupérer les valeurs de type string de l'objet graphique assigné à l'instance de classe. Deux signatures existent :

Récupérer la valeur de la propriété sans en vérifier l'exactitude

```
string  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,          // Identifiant de la propriété de typ
   int                          modifier=-1      // Modificateur
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type string de l'objet.

modifier=-1

[in] Modificateur (index) de la propriété de type string.

Valeur de Retour

Valeur de la propriété de type string.

Récupérer la valeur de la propriété en vérifiant son exactitude

```
bool  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,       // Identifiant de la propriété de type s
   int                          modifier,      // Modificateur
   string&                      value          // Référence vers la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type string de l'objet.

modifier

[in] Modificateur (index) de la propriété de type string.

value

[out] Référence à une variable permettant de récupérer la valeur de la propriété.

Valeur de Retour

vrai si réalisé avec succès, faux si l'objet ne peut pas renvoyer une valeur de type string.

Exemple :

```
//--- exemple d'utilisation de CChartObject::GetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
```

© 2000-2015, MetaQuotes Software Corp.

```
   CChartObject object;
   string       value;
   //--- récupère le nom de l'objet par la méthode simple
   printf("Object name is '%s'",object.GetString(OBJPROP_NAME));
   //--- récupère le nom de l'objet par la méthode classique
   if(!object.GetString(OBJPROP_NAME,0,value))
     {
      printf("Get string property error %d",GetLastError());
      return;
     }
   else
      printf("Object name is '%s'",value);
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- récupère la description des niveaux par la méthode simple
      printf("Level %d description is '%s'",i,object.GetString(OBJPROP_LEVELTEXT,i));
      //--- récupère la description des niveaux par la méthode classique
      if(!object.GetString(OBJPROP_LEVELTEXT,i,value))
        {
         printf("Get string property error %d",GetLastError());
         return;
        }
      else
         printf("Level %d description is '%s'",i,value);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

SetString

Fournit un accès simplifié aux fonctions de l'API MQL5 ObjectSetString() pour changer les propriétés de type string lié à une instance de la classe de l'objet graphique. Deux signatures existent :

Définition de la valeur d'une propriété ne nécessitant pas de modificateur

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,     // Identifiant de la propriété de type s
   string                        value        // Vaeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type string de l'objet.

value

[in] Nouvelle valeur de la propriété de type string et mutable.

Définition de la valeur d'une propriété en indiquant un modificateur

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,      // Identifiant de la propriété de type
   int                           modifier,     // Modificateur
   string                        value         // Valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété de type string de l'objet.

modifier

[in] Modificateur (index) de la propriété de type string.

value

[in] Nouvelle valeur de la propriété de type string et mutable.

Valeur de Retour

vrai si réalisé avec succès, faux si la propriété de type string ne peut pas être modifiée.

Exemple :

```
//--- exemple d'utilisation de CChartObject::SetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- définit le nouveau nom de l'objet graphique
   if(!object.SetString(OBJPROP_NAME,"MyObject"))
```

© 2000-2015, MetaQuotes Software Corp.

```
    {
     printf("Set string property error %d",GetLastError());
     return;
    }
  for(int i=0;i<object.LevelsCount();i++)
    {
     //--- définit la desription des niveaux
     if(!object.SetString(OBJPROP_LEVELTEXT,i,"Level_"+IntegerToString(i)))
       {
        printf("Set string property error %d",GetLastError());
        return;
       }
    }
}
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Save
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CChartObject object=new CChartObject;
   //--- définit les paramètres de l'objet
   //--- . . .
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres de l'objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de Retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Load
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CChartObject object;
   //--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- utilise l'objet
   //--- . . .
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type d'un objet graphique.

```
virtual int  Type() const
```

Valeur de Retour

Identifiant du type de l'objet (0x8888 pour CChartObject).

Exemple :

```
//--- exemple d'utilisation de CChartObject::Type
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- récupère le type de l'objet
   int type=object.Type();
  }
```

© 2000-2015, MetaQuotes Software Corp.

Objets Lignes

Groupe d'objets graphiques de type "Lignes".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Lignes" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectVLine	Objet graphique "Ligne Verticale"
CChartObjectHLine	Objet graphique "Ligne Horizontale"
CChartObjectTrend	Objet graphique "Ligne de Tendance"
CChartObjectTrendByAngle	Objet graphique "Ligne de Tendance par Angle"
CChartObjectCycles	Objet graphique "Ligne de Cycles"

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

CChartObjectVLine

La classe CChartObjectVLine permet un accès simplifié aux propriétés de l'objet graphique "Ligne Verticale".

Description

La classe CChartObjectVLine fournit un accès aux propriétés de l'objet "Ligne Verticale".

Déclaration

```
class CChartObjectVLine : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Ligne Verticale"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Ligne Verticale".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time           // Coordonnée date/heure
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time

[in] Coordonnée date/heure du point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_VLINE pour CChartObjectVLine).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectHLine

La classe CChartObjectHLine permet un accès simplifié aux propriétés de l'objet graphique "Ligne Horizontale".

Description

La classe CChartObjectHLine fournit un accès aux propriétés de l'objet "Ligne Horizontale".

Déclaration

```
class CChartObjectHLine : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Ligne Horizontale"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Ligne Horizontale".

```
bool  Create(
   long    chart_id,      // Identifiant du graphique
   string  name,          // Nom de l'objet
   long    window,        // Fenêtre du graphique
   double  price          // Coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

price

[in] Coordonnée prix du point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_HLINE pour CChartObjectHLine).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectTrend

La classe CChartObjectTrend permet un accès simplifié aux propriétés de l'objet graphique "Ligne de Tendance".

Description

La classe CChartObjectTrend fournit un accès aux propriétés de l'objet "Ligne de Tendance".

Déclaration

```
class CChartObjectTrend : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Ligne de Tendance"
Propriétés	
RayLeft	Retourne/Définit la propriété "Ray Left"
RayRight	Retourne/Définit la propriété "Ray Right"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Classes dérivées :

- CChartObjectChannel
- CChartObjectFibo
- CChartObjectFiboChannel
- CChartObjectFiboExpansion
- CChartObjectGannFan
- CChartObjectGannGrid
- CChartObjectPitchfork
- CChartObjectRegression
- CChartObjectStdDevChannel

© 2000-2015, MetaQuotes Software Corp.

- CChartObjectTrendByAngle

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Ligne de Tendance".

```
bool  Create(
   long      chart_id,     // Identifiant du graphique
   string    name,         // Nom de l'objet
   int       window,       // Fenêtre du graphique
   datetime  time1,        // Première coordonnée date/heure
   double    price1,       // Première coordonnée prix
   datetime  time2,        // Seconde coordonnée date/heure
   double    price2        // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

RayLeft (Méthode "Get")

Retourne la valeur de la propriété "Ray Left".

```
bool  RayLeft() const
```

Valeur de retour

Valeur de la propriété "Ray Left" assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

RayLeft (Méthode "Set")

Définit la nouvelle valeur de la propriété "Ray Left".

```
bool  RayLeft(
   bool  ray      // flag
   )
```

Paramètres

ray

[in] Nouvelle valeur de la propriété "Ray Left".

Valeur de retour

vrai si réalisé avec succès, faux si le flag n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

RayRight (Méthode "Get")

Retourne la valeur de la propriété "Ray Right".

```
bool  RayRight() const
```

Valeur de retour

Valeur de la propriété "Ray Right" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

RayRight (Méthode "Set")

Définit la nouvelle valeur de la propriété "Ray Right".

```
bool  RayRight(
   bool  ray      // flag
   )
```

Paramètres

ray

[in] Nouvelle valeur de la propriété "Ray Right".

Valeur de retour

vrai si réalisé avec succès, faux si le flag n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier précédemment ouvert en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_TREND pour CChartObjectTrend).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectTrendByAngle

La classe CChartObjectTrendByAngle fournit un accès simplifié aux propriétés de l'objet graphique "Ligne de Tendance par Angle".

Description

La classe CChartObjectTrendByAngle fournit un accès aux propriétés de l'objet "Ligne de Tendance par Angle".

Déclaration

```
class CChartObjectTrendByAngle : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Ligne de Tendance par Angle"
Propriétés	
Angle	Retourne/Définit la propriété "Angle"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Classes dérivées :

- CChartObjectGannLine

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Ligne de Tendance par Angle"

```
bool  Create(
   long      chart_id,     // Identifiant du graphique
   string    name,         // Nom de l'objet
   long      window,       // Fenêtre du graphique
   datetime  time1,        // Première coordonnée date/heure
   double    price1,       // Première coordonnée prix
   datetime  time2,        // Seconde coordonnée date/heure
   double    price2        // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Angle (Méthode "Get")

Retourne la valeur de la propriété "Angle".

```
double  Angle() const
```

Valeur de retour

Valeur de la propriété "Angle" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

Angle (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Angle".

```
bool  Angle(
   double  angle      // Angle
   )
```

Paramètres

angle

[in] Nouvelle valeur pour la propriété "Angle".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_TRENDBYANGLE pour CChartObjectTrendByAngle).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectCycles

La classe CChartObjectCycles permet un accès simplifié aux propriétés de l'objet graphique "Lignes de Cycles" .

Description

La classe CChartObjectCycles fournit un accès aux propriétés de l'objet "Lignes de Cycles".

Déclaration

```
class CChartObjectCycles : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Lignes de Cycles"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Lignes de Cycles"

```
bool  Create(
   long      chart_id,     // Identifiant du graphique
   string    name,         // Nom de l'objet
   long      window,       // Fenêtre du graphique
   datetime  time1,        // Première coordonnée date/heure
   double    price1,       // Première coordonnée prix
   datetime  time2,        // Seconde coordonnée date/heure
   double    price2        // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_CYCLES pour CChartObjectCycles).

© 2000-2015, MetaQuotes Software Corp.

Objets Canaux

Groupe d'objets graphiques de type "Canaux".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Canaux" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectChannel	Objet graphique "Canal Equidistant"
CChartObjectRegression	Objet graphique "Canal de Régression Linéaire"
CChartObjectStdDevChannel	Objet graphique "Canal de Déviation Standard"
CChartObjectPitchfork	Objet graphique "Fourchette d'Andrews"

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

CChartObjectChannel

La classe CChartObjectChannel permet un accès simplifié aux propriétés de l'objet graphique "Canal Equidistant".

Description

La classe CChartObjectChannel fournit un accès aux propriétés de l'objet "Canal Equidistant".

Déclaration

```
class CChartObjectChannel : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Canal Equidistant"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Canal Equidistant"

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Coordonnée date/heure du premier point d'ancrage
   double    price1,        // Coordonnée prix du premier point d'ancrage
   datetime  time2,         // Coordonnée date/heure du deuxième point d'ancrage
   double    price2,        // Coordonnée prix du second point d'ancrage
   datetime  time3,         // Coordonnée date/heure du troisième point d'ancrage
   double    price3         // Coordonnée prix du troisième point d'ancrage
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_CHANNEL pour CChartObjectChannel).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectRegression

La classe CChartObjectRegression permet un accès simplifié aux propriétés de l'objet graphique "Canal de Régression Linéaire".

Description

La classe CChartObjectRegression fournit un accès aux propriétés de l'objet "Canal de Régression Linéaire".

Déclaration

```
class CChartObjectRegression : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique "Canal de Régression Linéaire"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique "Canal de Régression Linéaire"

```
bool  Create(
   long      chart_id,     // Identifiant du graphique
   string    name,         // Nom de l'objet
   long      window,       // Fenêtre du graphique
   datetime  time1,        // Première coordonnée date/heure
   datetime  time2         // Seconde coordonnée date/heure
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_REGRESSION pour CChartObjectRegression).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectStdDevChannel

La class CChartObjectStdDevChannel permet un accès simplifié aux propriétés de l'objet graphique "Canal de Déviation Standard".

Description

La classe CChartObjectStdDevChannel fournit un accès aux propriétés de l'objet "Canal de Déviation Standard".

Déclaration

```
class CChartObjectStdDevChannel : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Canal de Déviation Standard"
Propriétés	
Deviation	Retourne/Définit la propriété "Deviation"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Canal de Déviation Standard"

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   datetime  time2,         // Seconde coordonnée date/heure
   double    deviation      // Déviation
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

deviation

[in] Valeur numérique pour la propriété "Deviation".

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Deviation (Méthode "Get")

Retourne la valeur numérique de la propriété "Deviation".

```
double  Deviation() const
```

Valeur de retour

Valeur numérique de la propriété "Deviation", assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

Deviation (Méthode "Set")

Définit la valeur numérique de la propriété "Deviation".

```
bool  Deviation(
   double  deviation      // Déviation
   )
```

Paramètres

deviation

[in] Nouvelle valeur pour la propriété "Deviation".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_STDDEVCHANNEL pour CChartObjectStdDevChannel).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectPitchfork

La classe CChartObjectPitchfork permet un accès simplifié aux propriétés de l'objet graphique "Fourchette d'Andrews".

Description

La classe CChartObjectPitchfork fournit un accès aux propriétés de l'objet "Fourchette d'Andrews".

Déclaration

```
class CChartObjectPitchfork : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Fourchette d'Andrews"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Fourchette d'Andrews"

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   long      window,        // Fenêtre du graphique
   datetime  time1,         // Coordonnée date/heure du premier point d'ancrage
   double    price1,        // Coordonnée prix du premier point d'ancrage
   datetime  time2,         // Coordonnée date/heure du deuxième point d'ancrage
   double    price2,        // Coordonnée prix du second point d'ancrage
   datetime  time3,         // Coordonnée date/heure du troisième point d'ancrage
   double    price3         // Coordonnée prix du troisième point d'ancrage
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_PITCHFORK pour CChartObjectPitchfork).

© 2000-2015, MetaQuotes Software Corp.

Outils de Gann

Groupe d'objets graphiques de type "Outils de Gann".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Outils de Gann" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectGannLine	Objet graphique "Ligne de Gann"
CChartObjectGannFan	Objet graphique "Eventail de Gann"
CChartObjectGannGrid	Objet graphique "Grille de Gann"

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

CChartObjectGannLine

La classe CChartObjectGannLine permet un accès simplifié aux propriétés de l'objet graphique "Ligne de Gann".

Description

La classe CChartObjectGannLine fournit un accès aux propriétés de l'objet "Ligne de Gann".

Déclaration

```
class CChartObjectGannLine : public CChartObjectTrendByAngle
```

Titre

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Méthodes de Classe

Create	
Create	Crée un objet graphique de type "Ligne de Gann"
Propriétés	
PipsPerBar	Retourne/Définit la propriété "Scale"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Ligne de Gann".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    ppb            // Pips par barre
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

ppb

[in] Pips par barre.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

PipsPerBar (Méthode "Get")

Retourne la valeur de la propriété "Pips per bar".

```
double  PipsPerBar() const
```

Valeur de retour

Valeur de la propriété "Pips per bar" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

PipsPerBar (Méthode "Set")

Définit la nouvelle valeur de la propriété "Pips per bar".

```
bool  PipsPerBar(
   double  ppb      // Pips par barre
   )
```

Paramètres

ppb

[in] Nouvelle valeur de la propriété "Pips per bar".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_GANNLINE pour CChartObjectGannLine).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectGannFan

La classe CChartObjectGannFan permet un accès simplifié aux propriétés de l'objet graphique "Eventail de Gann".

Description

La classe CChartObjectGannFan fournit un accès aux propriétés de l'objet "Eventail de Gann".

Déclaration

```
class CChartObjectGannFan : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Eventail de Gann"
Propriétés	
PipsPerBar	Retourne/Définit la propriété "Pips per bar"
Downtrend	Retourne/Définit la propriété "Downtrend"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Eventail de Gann".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    ppb            // Pips par barre
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

ppb

[in] Pips par barre.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

PipsPerBar (Méthode "Get")

Retourne la valeur de la propriété "Pips per bar".

```
double  PipsPerBar() const
```

Valeur de retour

Valeur de la propriété "Pips per bar" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

PipsPerBar (Méthode "Set")

Définit la nouvelle valeur de la propriété "Pips per bar".

```
bool  PipsPerBar(
   double  ppb      // Pips par barre
   )
```

Paramètres

ppb

[in] Nouvelle valeur de la propriété "Pips per bar".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Downtrend (Méthode "Get")

Retourne la valeur de la propriété "Downtrend".

```
bool  Downtrend() const
```

Valeur de retour

Valeur de la propriété "Downtrend" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Downtrend (Méthode "Set")

Définit la nouvelle valeur de la propriété "Downtrend".

```
bool  Downtrend(
   bool  downtrend      // Valeur du flag
   )
```

Paramètres

downtrend

[in] Nouvelle valeur de la propriété "Downtrend".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_GANNFAN pour CChartObjectGannFan).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectGannGrid

La classe CChartObjectGannGrid permet un accès simplifié aux propriétés de l'objet graphique "Grille de Gann".

Description

La classe CChartObjectGannGrid fournit un accès aux propriétés de l'objet "Grille de Gann".

Déclaration

```
class CChartObjectGannGrid : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Grille de Gann"
Propriétés	
PipsPerBar	Retourne/Définit la propriété "Pips per bar"
Downtrend	Retourne/Définit la propriété "Downtrend"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Grille de Gann".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    ppb            // Pips par barre
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

ppb

[in] Pips par barre.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

PipsPerBar (Méthode "Get")

Retourne la valeur de la propriété "Pips per bar".

```
double  PipsPerBar() const
```

Valeur de retour

Valeur de la propriété "Pips per bar" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

PipsPerBar (Méthode "Set")

Définit la nouvelle valeur de la propriété "Pips per bar".

```
bool  PipsPerBar(
   double  ppb      // Pips par barre
   )
```

Paramètres

ppb

[in] Nouvelle valeur de la propriété "Pips per bar".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Downtrend (Méthode "Get")

Retourne la valeur de la propriété "Downtrend".

```
bool  Downtrend() const
```

Valeur de retour

Valeur de la propriété "Downtrend" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Downtrend (Méthode "Set")

Définit la nouvelle valeur de la propriété "Downtrend".

```
bool  Downtrend(
   bool  downtrend      // Valeur du flag
   )
```

Paramètres

downtrend

[in] Nouvelle valeur de la propriété "Downtrend".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_GANNGRID pour CChartObjectGannGrid).

© 2000-2015, MetaQuotes Software Corp.

Outils de Fibonacci

Groupe d'objets graphiques de type "Outils de Fibonacci".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Outils de Fibonacci" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectFibo	Objet graphique "Retracement de Fibonacci"
CChartObjectFiboTimes	Objet graphique "Zones de Temps de Fibonacci"
CChartObjectFiboFan	Objet graphique "Eventail de Fibonacci"
CChartObjectFiboArc	Objet graphique "Arc de Fibonacci"
CChartObjectFiboChannel	Objet graphique "Canal de Fibonacci"
CChartObjectFiboExpansion	Objet graphique "Expansion de Fibonacci"

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

CChartObjectFibo

La classe CChartObjectFibo permet un accès simplifié aux propriétés de l'objet graphique "Retracement de Fibonacci".

Description

La classe CChartObjectFibo fournit un accès aux propriétés de l'objet "Retracement de Fibonacci".

Déclaration

```
class CChartObjectFibo : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Méthodes de Classe

Create	
Create	Crée un objet graphique de type "Retracement de Fibonacci"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Retracement de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2         // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_FIBO pour CChartObjectFibo).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectFiboTimes

La classe CChartObjectFiboTimes permet un accès simplifié aux propriétés de l'objet graphique "Zones de Temps de Fibonacci".

Description

La classe CChartObjectFiboTimes fournit un accès aux propriétés de l'objet "Zones de Temps de Fibonacci".

Déclaration

```
class CChartObjectFiboTimes : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Zones de Temps de Fibonacci"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Zones de Temps de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2         // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_FIBOTIMES pour CChartObjectFiboTimes).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectFiboFan

La classe CChartObjectFiboFan permet un accès simplifié aux propriétés de l'objet graphique "Eventail de Fibonacci".

Description

La classe CChartObjectFiboFan fournit un accès aux propriétés de l'objet "Eventail de Fibonacci".

Déclaration

```
class CChartObjectFiboFan : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Méthodes de Classe

Create	
Create	Crée un objet graphique de type "Eventail de Fibonacci"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Eventail de Fibonacci".

```
bool  Create(
   long      chart_id,     // Identifiant du graphique
   string    name,         // Nom de l'objet
   int       window,       // Fenêtre du graphique
   datetime  time1,        // Première coordonnée date/heure
   double    price1,       // Première coordonnée prix
   datetime  time2,        // Seconde coordonnée date/heure
   double    price2        // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_FIBOFAN pour CChartObjectFiboFan).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectFiboArc

La classe CChartObjectFiboArc permet un accès simplifié aux propriétés de l'objet graphique "Arc de Fibonacci".

Description

La classe CChartObjectFiboArc fournit un accès aux propriétés de l'objet "Arc de Fibonacci".

Déclaration

```
class CChartObjectFiboArc : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Arc de Fibonacci"
Propriétés	
Scale	Retourne/Définit la propriété "Scale"
Ellipse	Retourne/Définit la propriété "Ellipse"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Arc de Fibonacci"

```
bool  Create(
   long      chart_id,     // Identifiant du graphique
   string    name,         // Nom de l'objet
   int       window,       // Fenêtre du graphique
   datetime  time1,        // Première coordonnée date/heure
   double    price1,       // Première coordonnée prix
   datetime  time2,        // Seconde coordonnée date/heure
   double    price2,       // Seconde coordonnée prix
   double    scale         // Echelle
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

scale

[in] Echelle.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Scale (Méthode "Get")

Retourne la valeur de la propriété "Scale".

```
double  Scale() const
```

Valeur de retour

Valeur de la propriété "Scale" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

Scale (Méthode "Set")

Définit une nouvelle valeur à la propriété "Scale".

```
bool  Scale(
   double  scale      // Echelle
   )
```

Paramètres

scale

[in] Nouvelle valeur pour la propriété "Scale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Ellipse (Méthode "Get")

Retourne la valeur de la propriété "Ellipse".

```
bool  Ellipse() const
```

Valeur de retour

Valeur de la propriété "Ellipse" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Ellipse (Méthode "Set")

Définit la nouvelle valeur du flag de la propriété "Ellipse".

```
bool  Ellipse(
   bool  ellipse      // valeur du flag
   )
```

Paramètres

ellipse

[in] Nouvelle valeur pour la propriété "Scale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_FIBOARC pour CChartObjectFiboArc).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectFiboChannel

La classe CChartObjectFiboChannel permet un accès simplifié aux propriétés de l'objet graphique "Canal de Fibonacci".

Description

La classe CChartObjectFiboChannel fournit un accès aux propriétés de l'objet "Canal de Fibonacci".

Déclaration

```
class CChartObjectFiboChannel : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Méthodes de Classe

Create	
Create	Crée un objet graphique de type "Canal de Fibonacci"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Canal de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   datetime  time3,         // Troisième coordonnée date/heure
   double    price3         // Troisième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_FIBOCHANNEL pour CChartObjectFiboChannel).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectFiboExpansion

La classe CChartObjectFiboExpansion permet un accès simplifié aux propriétés de l'objet graphique "Expansion de Fibonacci".

Description

La classe CChartObjectFiboExpansion fournit un accès aux propriétés de l'objet "Expansion de Fibonacci".

Déclaration

```
class CChartObjectFiboExpansion : public CChartObjectTrend
```

Titre

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Expansion de Fibonacci"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Expansion de Fibonacci".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   datetime  time3,         // Troisième coordonnée date/heure
   double    price3         // Troisième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_EXPANSION pour CChartObjectFiboExpansion).

© 2000-2015, MetaQuotes Software Corp.

Outils d'Elliott

Groupe d'objets graphiques de type "Outils d'Elliott".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Outils d'Elliott" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectElliottWave3	Objet graphique "Vague de Correction"
CChartObjectElliottWave5	Objet graphique "Vague d'Impulsion"

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

CChartObjectElliottWave3

La classe CChartObjectElliottWave3 permet un accès smplifié aux propriétés de l'objet graphique "Vague de Correction".

Description

La classe CChartObjectElliottWave3 fournit un accès aux propriétés de l'objet "Vague de Correction".

Déclaration

```
class CChartObjectElliottWave3 : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Vague de Correction"
Propriétés	
Degree	Retourne/Définit la propriété "Degree"
Lignes	Retourne/Définit la propriété "Lines"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Classes dérivées :

- CChartObjectElliottWave5

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Vague de Correction".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   datetime  time3,         // Troisième coordonnée date/heure
   double    price3         // Troisième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée date/heure du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Degree (Méthode "Get")

Retourne la valeur de la propriété "Degree".

```
ENUM_ELLIOT_WAVE_DEGREE  Degree() const
```

Valeur de retour

Valeur de la propriété "Degree" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Degree (Méthode "Set")

Définit la nouvelle valeur de la propriété "Degree".

```
bool  Degree(
   ENUM_ELLIOT_WAVE_DEGREE  degree      // valeur de la propriété
   )
```

Paramètres

degree

[in] Nouvelle valeur de la propriété "Degree".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Lines (Méthode "Get")

Retourne la valeur de la propriété "Lines".

```
bool  Lines() const
```

Valeur de retour

Valeur de la propriété "Lines" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Lines (Méthode "Set")

Définit la nouvelle valeur de la propriété "Lines".

```
bool  Lines(
   bool  lines      // valeur du flag
   )
```

Paramètres

lines

[in] Nouvelle valeur de la propriété "Lines".

Valeur de retour

vrai si réalisé avec succès, faux si le flag n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_ELLIOTWAVE3 pour CChartObjectElliottWave3).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectElliottWave5

La classe CChartObjectElliottWave5 permet un accès simplifié aux propriété de l'objet graphique "Vague d'Impulsion".

Description

La classe CChartObjectElliottWave5 fournit un accès aux propriétés de l'objet "Vague d'Impulsion".

Déclaration

```
class CChartObjectElliottWave5 : public CChartObjectElliottWave3
```

Titre

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Vague d'Impulsion"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Vague d'Impulsion".

```
bool  Create(
   long      chart_id,     // Identifiant du graphique
   string    name,         // Nom de l'objet
   int       window,       // Fenêtre du graphique
   datetime  time1,        // Première coordonnée date/heure
   double    price1,       // Première coordonnée prix
   datetime  time2,        // Seconde coordonnée date/heure
   double    price2,       // Seconde coordonnée prix
   datetime  time3,        // Troisième coordonnée date/heure
   double    price3,       // Troisième coordonnée prix
   datetime  time4,        // Quatrième coordonnée date/heure
   double    price4,       // Quatrième coordonnée prix
   datetime  time5         // Cinquième coordonnée date/heure
   double    price5,       // Cinquième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

time4

[in] Coordonnée date/heure du quatrième point d'ancrage.

© 2000-2015, MetaQuotes Software Corp.

price4

[in] Coordonnée prix du quatrième point d'ancrage.

time5

[in] Coordonnée date/heure du cinquième point d'ancrage.

price5

[in] Coordonnée prix du cinquième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_ELLIOTWAVE5 pour CChartObjectElliottWave5).

© 2000-2015, MetaQuotes Software Corp.

Objets Formes

Groupe d'objets graphiques de type "Formes".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Formes" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectRectangle	Objet graphique "Rectangle"
CChartObjectTriangle	Objet graphique "Triangle"
CChartObjectEllipse	Objet graphique "Ellipse"

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

CChartObjectRectangle

La classe CChartObjectRectangle permet un accès simplifié aux propriétés de l'objet graphique "Rectangle Rectangle".

Description

La classe CChartObjectRectangle fournit un accès aux propriétés de l'objet "Rectangle".

Déclaration

```
S   class CChartObjectRectangle : public CChartObject
```

Titre

```
   #include <ChartObjects\ChartObjectsShapes.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Rectangle"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Rectangle".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   long      window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2         // Seconde coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_RECTANGLE pour CChartObjectRectangle).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectTriangle

La classe CChartObjectTriangle permet un accès simplifié aux propriétés de l'objet graphique "Triangle".

Description

La classe CChartObjectTriangle fournit un accès aux propriétés de l'objet "Triangle".

Déclaration

```
class CChartObjectTriangle : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Triangle"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Triangle".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   long      window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   datetime  time3,         // Troisième coordonnée date/heure
   double    price3         // Troisième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_TRIANGLE pour CChartObjectTriangle).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectEllipse

La classe CChartObjectEllipse permet un accès simplifié aux propriétés de l'objet graphique "Ellipse".

Description

La classe CChartObjectEllipse fournit un accès aux propriétés de l'objet "Ellipse".

Déclaration

```
class CChartObjectEllipse : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Ellipse"
Entrée/Sortie	
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Ellipse".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time1,         // Première coordonnée date/heure
   double    price1,        // Première coordonnée prix
   datetime  time2,         // Seconde coordonnée date/heure
   double    price2,        // Seconde coordonnée prix
   datetime  time3,         // Troisième coordonnée date/heure
   double    price3         // Troisième coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time1

[in] Coordonnée date/heure du premier point d'ancrage.

price1

[in] Coordonnée prix du premier point d'ancrage.

time2

[in] Coordonnée date/heure du deuxième point d'ancrage.

price2

[in] Coordonnée prix du second point d'ancrage.

time3

[in] Coordonnée date/heure du troisième point d'ancrage.

price3

[in] Coordonnée prix du troisième point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_ELLIPSE pour CChartObjectEllipse).

© 2000-2015, MetaQuotes Software Corp.

Objets Flèches

Groupe d'objets graphiques Flèches.

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Flèches" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library). Par définition, une flèche est une icone à afficher à l'écran et utilisant un certain code. Il existe 2 types d'objets graphiques de type "Flèches" permettant d'afficher des icones sur les graphiques :

- L'objet "Flèche", vous permettant de spécifier le code de l'icone à afficher.
- Les groupes d'objets pour afficher certains types d'icones (et le code fixé correspondant).

Classe pour l'affichage d'icones avec un code arbitraire

Nom de la classe	Nom de l'objet flèche
CChartObjectArrow	Arrow

Classes pour l'affichage d'icones prédéfinies (code fixé)

Nom de la classe	Nom de l'objet flèche
CChartObjectArrowCheck	Coche
CChartObjectArrowDown	Flèche vers le Haut
CChartObjectArrowUp	Flèche vers le Bas
CChartObjectArrowStop	Signe Stop
CChartObjectArrowThumbDown	Pouce vers le Haut
CChartObjectArrowThumbUp	Pouce vers le Bas
CChartObjectArrowLeftPrice	Etiquette Prix à Gauche
CChartObjectArrowRightPrice	Etiquette Prix à Droite

Voir aussi

Types des objets, Points d'ancrage des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

CChartObjectArrow

La classe CChartObjectArrow permet un accès simplifié aux propriétés de l'objet graphique 'Flèche".

Description

La classe CChartObjectArrow fournit un accès aux propriétés générales des objets 'Flèches" à tous ses descendants.

Déclaration

```
class CChartObjectArrow : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsArrows.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique "Flèche"
Propriétés	
ArrowCode	Retourne/Définit la propriété "Arrow Code"
Anchor	Retourne/Définit la propriété "Anchor"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Méthodes d'accrochage des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Flèche".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Numéro de fenêtre
   datetime  time,          // Date/heure
   double    price,         // Prix
   char      code           // Code de flèche
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Nom de l'objet (doit être unique).

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time

[in] Date/heure.

price

[in] Prix.

code

[in] Code de la "Flèche" (Wingdings).

Valeur de retour

vrai - en cas de succès, faux sinon

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrow::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- erreur de création de la flèche
      printf("Arrow create: Error %d!",GetLastError());
      //---
```

© 2000-2015, MetaQuotes Software Corp.

```
      return;
     }
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2015, MetaQuotes Software Corp.

ArrowCode (Méthode "Get")

Retourne le code du symbole de l'objet "Flèche".

```
char  ArrowCode() const
```

Valeur de retour

Code du symbole utilisé par l'objet "Flèche", assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

ArrowCode (Méthode "Set")

Définit le code du symbole à utiliser par l'objet "Flèche"

```
bool  ArrowCode(
   char  code      // Valeur du code
   )
```

Paramètres

code

[in] nouveau code du symbole à utiliser par l'objet "Flèche" (Wingdings).

Valeur de retour

vrai si réalisé avec succès, faux si le code du symbole n'a pas été changé.

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrow::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   char              code=181;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,code))
     {
      //--- erreur de création de la flèche
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- vous pouvez maintenant utiliser la Flèche
//--- . . .
//--- récupère le code du symbole de la "Flèche"
   if(arrow.ArrowCode()!=code)
     {
     //--- définit le code de la "Flèche"
```

© 2000-2015, MetaQuotes Software Corp.

```
      arrow.ArrowCode(code);
      }
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2015, MetaQuotes Software Corp.

Anchor (Méthode "Get")

Retourne le type de point d'ancrage de l'objet "Flèche"

```
ENUM_ARROW_ANCHOR  Anchor() const
```

Valeur de retour

Type d'ancrage de l'objet "Flèche", assigné à l'instance de classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Anchor (Méthode "Set")

Définit le type de point d'ancrage de l'objet "Flèche"

```
bool  Anchor(
   ENUM_ARROW_ANCHOR  anchor      // nouveau type d'accrochage
   )
```

Paramètres

anchor

[in] Nouveau type d'accrochage

Valeur de retour

vrai si réalisé avec succès, faux si le type d'accrochage n'a pas été changé.

Exemple :

```
//--- exemple d'utilisation de CChartObject::Anchor
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- erreur de création de la flèche
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- retourne l'ancrage de la flèche
   if(arrow.Anchor()!=anchor)
     {
     //--- définit l'ancrage de la flèche
     arrow.Anchor(anchor);
     }
```

© 2000-2015, MetaQuotes Software Corp.

```
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai - en cas de succès, faux sinon

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrow::Save
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- erreur de création de la flèche
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Save(file_handle))
        {
         //--- erreur de sauvegarde du fichier
         printf("File save: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier ouvert précédemment en utilisant la fonction FileOpen(...).

Valeur de retour

vrai - en cas de succès, faux sinon

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrow::Load
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- ouvre le fichier
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Load(file_handle))
        {
         //--- erreur de chargement du fichier
         printf("File load: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (par exemple, OBJ_ARROW pour CChartObjectArrow)

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrow::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   //--- récupère le type de flèche
   int type=arrow.Type();
  }
```

© 2000-2015, MetaQuotes Software Corp.

Flèches prédéfinies

Les classes "Flèches prédéfinies" permettent un accès simplifié aux propriété des objets graphiques suivants :

Nom de la classe	Nom de l'objet Flèche
CChartObjectArrowCheck	"Coche"
CChartObjectArrowDown	"Flèche vers le Bas"
CChartObjectArrowUp	"Flèche vers le Haut"
CChartObjectArrowStop	"Flèche Stop"
CChartObjectArrowThumbDown	"Correct" ("Pouce vers le haut")
CChartObjectArrowThumbUp	"Mauvais" ("Pouce vers le bas")
CChartObjectArrowLeftPrice	Flèche "Prix à gauche"
CChartObjectArrowRightPrice	Flèche "Prix à droite"

Description

Les classes "Flèches prédéfinies" fournissent l'accès aux propriété de l'objet.

Déclarations

```
class CChartObjectArrowCheck      : public CChartObjectArrow;
class CChartObjectArrowDown       : public CChartObjectArrow;
class CChartObjectArrowUp         : public CChartObjectArrow;
class CChartObjectArrowStop       : public CChartObjectArrow;
class CChartObjectArrowThumbDown  : public CChartObjectArrow;
class CChartObjectArrowThumbUp    : public CChartObjectArrow;
class CChartObjectArrowLeftPrice  : public CChartObjectArrow;
class CChartObjectArrowRightPrice : public CChartObjectArrow;
```

Titre

```
<ChartObjects\ChartObjectsArrows.mqh>
```

Méthodes de Classe

Création	
Create	Crée l'objet graphique spécifié
Propriétés	
ArrowCode	</t7><li8>"Stub" pour la méthode de changement de code
Entrée/Sortie	

© 2000-2015, MetaQuotes Software Corp.

virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Méthodes d'accrochage des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Flèche prédéfinie".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Numéro de fenêtre
   datetime  time,          // Date/heure
   double    price          // Prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time

[in] Date/heure.

price

[in] Prix.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrowCheck::Create
//--- exemple d'utilisation de CChartObjectArrowDown::Create
//--- exemple d'utilisation de CChartObjectArrowUp::Create
//--- exemple d'utilisation de CChartObjectArrowStop::Create
//--- exemple d'utilisation de CChartObjectArrowThumbDown::Create
//--- exemple d'utilisation de CChartObjectArrowThumbUp::Create
//--- exemple d'utilisation de CChartObjectArrowLeftPrice::Create
//--- exemple d'utilisation de CChartObjectArrowRightPrice::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- par exemple, prenez CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
```

© 2000-2015, MetaQuotes Software Corp.

```
      {
       //--- erreur de création de la flèche
       printf("Arrow create: Error %d!",GetLastError());
       //---
       return;
      }
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2015, MetaQuotes Software Corp.

ArrowCode

Interdit le changement du code des flèches.

```
bool  ArrowCode(
   char  code      // valeur du code
   )
```

Paramètres

code

[in] n'importe quelle valeur

Valeur de retour

Toujours faux.

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrowCheck::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowDown::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowUp::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowStop::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowThumbDown::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowThumbUp::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowLeftPrice::ArrowCode
//--- exemple d'utilisation de CChartObjectArrowRightPrice::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- par exemple, prenez CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- définit les paramètres de l'objet
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
     {
      //--- erreur de création de la flèche
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- définit le code de la flèche
   if(!arrow.ArrowCode(181))
     {
      //--- ce n'est pas une erreur
      printf("Arrow code can not be changed");
     }
//--- vous pouvez maintenant utiliser la flèche
//--- . . .
  }
```

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_ARROW_CHECK pour CChartObjectArrowCheck, OBJ_ARROW_DOWN pour CChartObjectArrowDown, OBJ_ARROW_UP pour CChartObjectArrowUp, OBJ_ARROW_STOP pour CChartObjectArrowStop, OBJ_ARROW_THUMB_DOWN pour CChartObjectArrowThumbDown, OBJ_ARROW_THUMB_UP pour CChartObjectArrowThumbUp, OBJ_ARROW_LEFT_PRICE pour CChartObjectArrowLeftPrice, OBJ_ARROW_RIGHT_PRICE pour CChartObjectArrowRightPrice).

Exemple :

```
//--- exemple d'utilisation de CChartObjectArrowCheck::Type
//--- exemple d'utilisation de CChartObjectArrowDown::Type
//--- exemple d'utilisation de CChartObjectArrowUp::Type
//--- exemple d'utilisation de CChartObjectArrowStop::Type
//--- exemple d'utilisation de CChartObjectArrowThumbDown::Type
//--- exemple d'utilisation de CChartObjectArrowThumbUp::Type
//--- exemple d'utilisation de CChartObjectArrowLeftPrice::Type
//--- exemple d'utilisation de CChartObjectArrowRightPrice::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- par exemple, prenez CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- récupère le type de flèche
   int type=arrow.Type();
  }
```

© 2000-2015, MetaQuotes Software Corp.

Objet Contrôle

Groupe d'objets graphiques de type "Contrôle".

Cette section contient les détails techniques d'utilisation d'un groupe de classes d'objets graphiques de type "Contrôles" ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Nom de la classe	Objet
CChartObjectText	Objet graphique "Texte"
CChartObjectLabel	Objet graphique "Etiquette de Texte"
CChartObjectEdit	Objet graphique "Zone d'édition"
CChartObjectButton	Objet graphique "Bouton"
CChartObjectSubChart	Objet graphique "Graphique"
CChartObjectBitmap	Objet graphique "Bitmap"
CChartObjectBmpLabel	Objet graphique "Etiquette de Bitmap"
CChartObjectRectLabel	Objet graphique "Etiquette Rectangle"

Voir aussi

Types des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

CChartObjectText

La classe CChartObjectText permet un accès simplifié aux propriétés de l'objet graphique "Texte".

Description

La classe CChartObjectText fournit un accès aux propriétés de l'objet "Texte".

Déclaration

```
class CChartObjectText : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Texte"
Propriétés	
Angle	Retourne/Définit la propriété "Angle"
Font	Retourne/Définit la propriété "Font"
FontSize	Retourne/Définit la propriété "FontSize"
Anchor	Retourne/Définit la propriété "Anchor"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Classes dérivées :

- CChartObjectLabel

Voir aussi

Assigne une chaîne de caractèresTypes des objets, Propriétés d'un objet, Méthodes de lien avec un objet, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Texte".

```
bool  Create(
   long      chart_id,     // Identifiant du graphique
   string    name,         // Nom de l'objet
   int       window,       // Fenêtre du graphique
   datetime  time,         // Coordonnée date/heure
   double    price         // Coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time

[in] Coordonnée date/heure du point d'ancrage.

price

[in] Coordonnée prix du point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Angle (Méthode "Get")

Retourne la valeur de la propriété "Angle".

```
double  Angle() const
```

Valeur de retour

Valeur de la propriété "Angle" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

Angle (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Angle".

```
bool  Angle(
   double  angle      // nouvelle valeur
   )
```

Paramètres

angle

[in] Nouvelle valeur pour la propriété "Angle".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Font (Méthode "Get")

Retourne la valeur de la propriété "Font".

```
string  Font() const
```

Valeur de retour

Valeur de la propriété "Font" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne "".

Font (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Font".

```
bool  Font(
   string  font      // nouvelle police de caractères
   )
```

Paramètres

font

[in] Nouvelle valeur pour la propriété "Font".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

FontSize (Méthode "Get")

Retourne la valeur de la propriété "FontSize".

```
int  FontSize() const
```

Valeur de retour

Valeur de la propriété "FontSize" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

FontSize (Méthode "Set")

Définit une nouvelle valeur pour la propriété "FontSize".

```
bool  FontSize(
   int  size      // nouvelle taille de la police de caractères
   )
```

Paramètres

size

[in] Nouvelle valeur pour la propriété "Font".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Anchor (Méthode "Get")

Retourne la valeur de la propriété "Anchor".

```
ENUM_ANCHOR_POINT  Anchor() const
```

Valeur de retour

Valeur de la propriété "Anchor" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Anchor (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Anchor".

```
bool  Anchor(
   ENUM_ANCHOR_POINT  anchor      // nouvelle valeur
   )
```

Paramètres

anchor

[in] Nouvelle valeur pour la propriété "Anchor".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_TEXT pour CChartObjectText).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectLabel

La classe CChartObjectLabel perme un accès simplifié aux propriété de l'objet graphique "Label".

Description

La classe CChartObjectLabel fournit un accès aux propriétés de l'objet "Label".

Déclaration

```
class CChartObjectLabel : public CChartObjectText
```

Titre

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Label"
Propriétés	
X_Distance	Retourne/Définit la propriété "X_Distance"
Y_Distance	Retourne/Définit la propriété "Y_Distance"
X_Size	Retourne/Définit la propriété "X_Size"
Y_Size	Retourne/Définit la propriété "Y_Size"
Corner	Retourne/Définit la propriété "Corner"
Time	"Stub" pour le changement de coordonnées date/heure
Price	"Stub" pour le changement de coordonnées prix
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Classes dérivées :

- CChartObjectEdit

Voir aussi

Types des objets, Propriétés d'un objet, Angle d'un graphique, Points d'ancrage des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Label".

```
bool  Create(
   long     chart_id,      // Identifiant du graphique
   string   name,          // Nom de l'objet
   int      window,        // Fenêtre du graphique
   int      X,             // Coordonnée X
   int      Y              // Coordonnée Y
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

X

[in] Coordonnée X.

Y

[in] Coordonnée Y.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

X_Distance (Méthode "Get")

Retourne la valeur de la propriété "X_Distance".

```
int  X_Distance() const
```

Valeur de retour

Valeur de la propriété "X_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

X_Distance (Méthode "Set")

Définit une nouvelle valeur pour la propriété "X_Distance".

```
bool  X_Distance(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur pour la propriété "X_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Y_Distance (Méthode "Get")

Retourne la valeur de la propriété "Y_Distance".

```
int  Y_Distance() const
```

Valeur de retour

Valeur de la propriété "Y_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

Y_Distance (Méthode "Set")

Définit la nouvelle valeur de la propriété "Y_Distance".

```
bool  Y_Distance(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur pour la propriété "Y_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

X_Size

Retourne la valeur de la propriété "X_Size".

```
int  X_Size() const
```

Valeur de retour

Valeur de la propriété "X_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

Y_Size

Retourne la valeur de la propriété "Y_Size".

```
int  Y_Size() const
```

Valeur de retour

Valeur de la propriété "Y_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

Corner (Méthode "Get")

Retourne la valeur de la propriété "Corner".

```
ENUM_BASE_CORNER  Corner() const
```

Valeur de retour

Valeur de la propriété "Corner" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Corner (Méthode "Set")

Définit une nouvelle pour la propriété "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // nouvelle valeur
   )
```

Paramètres

corner

[in] Nouvelle valeur pour la propriété "Corner".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Time

Interdit le changement de la coordonnée date/heure.

```
bool  Time(
   datetime  time      // n'importe quelle valeur
   )
```

Paramètres

time

[in] N'importe quelle valeur de type datetime.

Valeur de retour

toujours faux.

© 2000-2015, MetaQuotes Software Corp.

Price

Interdit le changement de la coordonnée prix.

```
bool  Price(
   double  price      // n'importe quelle valeur
   )
```

Paramètres

price

[in] N'importe quelle valeur de type double.

Valeur de retour

toujours faux.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant de type de l'objet (OBJ_LABEL pour CChartObjectLabel).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectEdit

La classe CChartObjectEdit permet un accès simplifié aux propriétés de l'objet graphique "Zone d'édition".

Description

La classe CChartObjectEdit fournit un accès aux propriétés de l'objet "Zone d'édition".

Déclaration

```
class CChartObjectEdit : public CChartObjectLabel
```

Titre

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Zone d'édition"
Propriétés	
TextAlign	Retourne/Definit la propriété "TextAlign"
X_Size	Retourne la propriété "X Size"
Y_Size	Retourne la propriété "Y Size"
BackColor	Retourne/Définit la propriété "Background Color"
BorderColor	Retourne/Définit la propriété "Border Color"
ReadOnly	Retourne/Définit la propriété "Read Only"
Angle	Retourne/Définit la propriété "Angle"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Classes dérivées :

- CChartObjectButton

Voir aussi

© 2000-2015, MetaQuotes Software Corp.

Types des objets, Propriétés d'un objet, Angle d'un graphique, Points d'ancrage des objets, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique "zone d'édition".

```
bool  Create(
   long     chart_id,      // Identifiant du graphique
   string   name,          // Nom de l'objet
   int      window,        // Fenêtre du graphique
   int      X,             // Coordonnée X
   int      Y,             // Coordonnée Y
   int      sizeX,         // Taille X
   int      sizeY          // Taille Y
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

X

[in] Coordonnée X.

Y

[in] Coordonnée Y.

sizeX

[in] Taille X.

sizeY

[in] Taille Y.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

TextAlign (Méthode "Get")

Retourne la valeur de la propriété "TextAlign" (mode d'alignement du texte).

```
ENUM_ALIGN_MODE  TextAlign() const
```

Valeur de retour

Valeur de la propriété "TextAlign" de l'objet assigné à l'instance de la classe.

TextAlign (Méthode "Set")

Définit la valeur de la propriété (mode d'alignement du texte).

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // nouvelle valeur
   )
```

Paramètres

align

[in] Nouvelle valeur de la propriété "TextAlign".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

X_Size

Définit une nouvelle valeur pour la propriété "X_Size".

```
bool  X_Size(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle valeur pour la propriété "X_Size".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Y_Size

Définit une nouvelle valeur pour la propriété "Y_Size".

```
bool  Y_Size(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle valeur pour la propriété "Y_Size".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

BackColor (Méthode "Get")

Retourne la valeur de la propriété "BackColor".

```
color  BackColor() const
```

Valeur de retour

Valeur de la propriété "BackColor" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne CLR_NONE.

BackColor (Méthode "Set")

Définit une nouvelle valeur pour la propriété "BackColor".

```
bool  BackColor(
   color  new_color      // nouvelle couleur d'arrière plan
   )
```

Paramètres

new_color

[in] Nouvelle valeur pour la propriété "BackColor".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

BorderColor (Méthode "Get")

Retourne la valeur de la propriété "Border Color".

```
color  BorderColor() const
```

Valeur de retour

Valeur de la propriété "Border Color" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne CLR_NONE.

BorderColor (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Border Color".

```
bool  BorderColor(
   color  new_color      // nouvelle couleur de bordure
   )
```

Paramètres

new_color

[in] Nouvelle couleur pour la propriété "Border Color".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ReadOnly (Méthode "Get")

Retourne la valeur de la propriété "Read Only".

```
bool  ReadOnly() const
```

Valeur de retour

Valeur de la propriété "Read Only" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

ReadOnly (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Read Only".

```
bool  ReadOnly(
   const  bool  flag      // nouvelel valeur
   )
```

Paramètres

flag

[in] Nouvelle valeur pour la propriété "Read Only".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Angle

Interdit le changement de la propriété "Angle".

```
bool  Angle(
   double  angle      // n'importe quelle valeur
   )
```

Paramètres

angle

[in] N'importe quelle valeur de type double.

Valeur de retour

toujours faux.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_EDIT pour CChartObjectEdit).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectButton

La classe CChartObjectButton permet un accès simplifié aux propriétés de l'objet graphique "Bouton".

Description

La classe CChartObjectButton fournit un accès aux propriétés de l'objet "Bouton".

Déclaration

```
class CChartObjectButton : public CChartObjectEdit
```

Titre

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Méthodes de Classe

Création	
Create	Hérité de la class CChartObjectEdit
Propriétés	
Etat	Retourne/Définit l'état du bouton (Appuyé/ Relâché)
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Propriétés d'un objet, Angle d'un graphique, Méthodes de lien avec un objet, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

State (Méthode "Get")

Retourne la valeur de la propriété "State".

```
bool  State() const
```

Valeur de retour

Valeur de la propriété "State" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

State (Méthode "Set")

Définit une nouvelle valeur pour la propriété "State".

```
bool  State(
   bool  state      // nouvelle valeur d'état
   )
```

Paramètres

X

[in] Nouvelle valeur pour la propriété "State".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_BUTTON pour CChartObjectButton).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectSubChart

La classe CChartObjectSubChart permet un accès simplifié aux propriétés de l'objet graphique "Graphique".

Description

La classe CChartObjectSubChart fournit un accès aux propriétés de l'objet "Graphique".

Déclaration

```
class CChartObjectSubChart : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectSubChart.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Graphique"
Propriétés	
X_Distance	Retourne/Définit la propriété "X_Distance"
Y_Distance	Retourne/Définit la propriété "Y_Distance"
Corner	Retourne/Définit la propriété "Corner"
X_Size	Retourne/Définit la propriété "X_Size"
Y_Size	Retourne/Définit la propriété "Y_Size"
Symbol	Retourne/Définit la propriété "Symbol"
Period	Retourne/Définit la propriété "Period"
Scale	Retourne/Définit la propriété "Scale"
DateScale	Retourne/Définit la propriété "Show date scale"
PriceScale	Retourne/Définit la propriété "Show price scale"
Time	"Stub" pour le changement de coordonnées date/heure
Price	"Stub" pour le changement de coordonnées prix
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier

© 2000-2015, MetaQuotes Software Corp.

virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Propriétés des objet, Angle du graphique, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Sous-Graphique".

```
bool  Create(
   long     chart_id,      // Identifiant du graphique
   string   name,          // Nom de l'objet
   int      window,        // Fenêtre du graphique
   int      X,             // Coordonnée X
   int      Y,             // Coordonnée Y
   int      sizeX,         // Taille X
   int      sizeY          // Taille Y
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

X

[in] Coordonnée X.

Y

[in] Coordonnée Y.

sizeX

[in] Taille X.

sizeY

[in] Taille Y.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

X_Distance (Méthode "Get")

Retourne la valeur de la propriété "X_Distance".

```
int  X_Distance() const
```

Valeur de retour

Valeur de la propriété "X_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

X_Distance (Méthode "Set")

Définit une nouvelle valeur pour la propriété "X_Distance".

```
bool  X_Distance(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur pour la propriété "X_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Y_Distance (Méthode "Get")

Retourne la valeur de la propriété "Y_Distance".

```
int  Y_Distance() const
```

Valeur de retour

Valeur de la propriété "Y_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

Y_Distance (Méthode "Set")

Définit la nouvelle valeur de la propriété "Y_Distance".

```
bool  Y_Distance(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur pour la propriété "Y_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Corner (Méthode "Get")

Retourne la valeur de la propriété "Corner".

```
ENUM_BASE_CORNER Corner() const
```

Valeur de retour

Valeur de la propriété "Corner" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Corner (Méthode "Set")

Définit une nouvelle pour la propriété "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // nouvelle valeur
   )
```

Paramètres

corner

[in] Nouvelle valeur pour la propriété "Corner".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

X_Size (Méthode "Get")

Retourne la valeur de la propriété "X_Size".

```
int  X_Size() const
```

Valeur de retour

Valeur de la propriété "X_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

X_Size (Méthode "Set")

Définit une nouvelle valeur pour la propriété "X_Size".

```
bool  X_Size(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur pour la propriété "X_Size".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Y_Size (Méthode "Get")

Retourne la valeur de la propriété "Y_Size".

```
int  Y_Size() const
```

Valeur de retour

Valeur de la propriété "Y_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

Y_Size (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Y_Size".

```
bool  Y_Size(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur pour la propriété "Y_Size".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Symbol (Méthode "Get")

Retourne la valeur de la propriété "Symbol".

```
string  Symbol() const
```

Valeur de retour

Valeur de la propriété "Symbol" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne "".

Symbol (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Symbol".

```
bool  Symbol(
   string  symbol      // nouveau symbole
   )
```

Paramètres

symbol

[in] Nouvelle valeur pour la propriété "Symbol".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Period (Méthode "Get")

Retourne la valeur de la propriété "Period".

```
int  Period() const
```

Valeur de retour

Valeur de la propriété "Period" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

Period (Méthode "Set")

Définit une nouvelle valeur pour la propriété "Period".

```
bool  Period(
   int  period      // nouvelle valeur
   )
```

Paramètres

period

[in] Nouvelle valeur pour la propriété "Period".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Scale (Méthode "Get")

Retourne la valeur de la propriété "Scale".

```
double  Scale() const
```

Valeur de retour

Valeur de la propriété "Scale" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

Scale (Méthode "Set")

Définit une nouvelle valeur à la propriété "Scale".

```
bool  Scale(
   double  scale      // nouvelle valeur
   )
```

Paramètres

scale

[in] Nouvelle valeur pour la propriété "Scale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

DateScale (Méthode "Get")

Retourne la valeur de la propriété "DateScale".

```
bool  DateScale() const
```

Valeur de retour

Valeur de la propriété "DateScale" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

DateScale (Méthode "Set")

Définit une nouvelle valeur pour la propriété "DateScale".

```
bool  DateScale(
   bool  scale      // nouvelle valeur
   )
```

Paramètres

scale

[in] Nouvelle valeur pour la propriété "DateScale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

PriceScale (Méthode "Get")

Retourne la valeur de la propriété "PriceScale".

```
bool  PriceScale() const
```

Valeur de retour

Valeur la propriété "PriceScale" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

PriceScale (Méthode "Set")

Définit une nouvelle valeur pour la propriété "PriceScale".

```
bool  PriceScale(
   bool  scale      // nouvelle valeur
   )
```

Paramètres

scale

[in] Nouvelle valeur pour la propriété "PriceScale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Time

Interdit le changement de la coordonnée date/heure.

```
bool  Time(
   datetime  time      // n'importe quelle valeur
   )
```

Paramètres

time

[in] N'importe quelle valeur de type datetime.

Valeur de retour

toujours faux.

© 2000-2015, MetaQuotes Software Corp.

Price

Interdit le changement de la coordonnée prix.

```
bool  Price(
   double  price      // n'importe quelle valeur
   )
```

Paramètres

price

[in] N'importe quelle valeur de type double.

Valeur de retour

toujours faux.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_CHART pour CChartObjectSubChart).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectBitmap

La classe CChartObjectBitmap permet un accès simplifié aux propriétés de l'objet "Bitmap".

Description

La classe CChartObjectBitmap fournit un accès aux propriétés de l'objet "Bitmap".

Déclaration

```
class CChartObjectBitmap : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "Bitmap"
Propriétés	
BmpFile	Retourne/Définit la propriété "BMP Filename"
X_Offset	Retourne/Définit la propriété "X_Offset"
Y_Offset	Retourne/Définit la propriété "Y_Offset"
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Propriétés de l'objet, Objet graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "Bitmap".

```
bool  Create(
   long      chart_id,      // Identifiant du graphique
   string    name,          // Nom de l'objet
   int       window,        // Fenêtre du graphique
   datetime  time,          // Coordonnée date/heure
   double    price          // Coordonnée prix
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

time

[in] Coordonnée date/heure du point d'ancrage.

price

[in] Coordonnée prix du point d'ancrage.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

BmpFile (Méthode "Get")

Retourne la valeur de la propriété "BmpFile".

```
string  BmpFile() const
```

Valeur de retour

Valeur de la propriété "BmpFile" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

BmpFile (Méthode "Set")

Définit la nouvelle valeur de la propriété "BmpFile".

```
bool  BmpFile(
   string  name      // nouveau nom de fichier
   )
```

Paramètres

name

[in] Nouvelle valeur de la propriété "BmpFile".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

X_Offset (Méthode "Get")

Retourne la valeur de la propriété "X_Offset" (la coordonnée X du coin supérieur gauche du rectangle visible de l'objet).

```
int  X_Offset() const
```

Valeur de retour

Valeur de la propriété "X_Offset" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

X_Offset (Méthode "Set")

Définit la nouvelle valeur de la propriété "X_Offset" (la coordonnée X du coin supérieur gauche du rectangle visible de l'objet). La valeur est définie en pixels relativement au coin suéprieur gauche de l'image originale.

```
bool  X_Offset(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur de la propriété "X_Offset".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Y_Offset (Méthode "Get")

Retourne la valeur de la propriété "Y_Offset" (la coordonnée Y du coin supérieur gauche du rectangle visible de l'objet).

```
int  Y_Offset() const
```

Valeur de retour

Valeur de la propriété "Y_Offset" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

Y_Offset (Méthode "Set")

Définit la nouvelle valeur de la propriété "Y_Offset" (la coordonnée Y du coin supérieur gauche du rectangle visible de l'objet). La valeur est définie en pixels relativement au coin suéprieur gauche de l'image originale.

```
bool  Y_Offset(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur de la propriété "Y_Offset".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_BITMAP pour CChartObjectBitmap).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectBmpLabel

La classe CChartObjectBmpLabel permet un accès simplifié aux propriétés de l'objet graphique "Bitmap Label".

Description

La classe CChartObjectBmpLabel fournit un accès aux propriétés de l'objet "Bitmap Label".

Déclaration

```
class CChartObjectBmpLabel : public CChartObject
```

Titre

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Méthodes de Classe

Création	
Create	Crée un objet graphique de type "BmpLabel"
Propriétés	
X_Distance	Retourne/Définit la propriété "X_Distance"
Y_Distance	Retourne/Définit la propriété "Y_Distance"
X_Offset	Retourne/Définit la propriété "X_Offset"
Y_Offset	Retourne/Définit la propriété "Y_Offset"
Corner	Retourne/Définit la propriété "Corner"
X_Size	Retourne/Définit la propriété "X_Size"
Y_Size	Retourne/Définit la propriété "Y_Size"
BmpFileOn	Retourne/Définit la propriété "BmpFileOn" utilisée lorsqu'un bouton est appuyé (On)
BmpFileOff	Retourne/Définit la propriété "BmpFileOff" utilisée lorsqu'un bouton est relâché (Off)
Etat	Retourne/Définit la propriété "Button State" (Appuyé/Relâché)
Time	"Stub" pour le changement de coordonnées date/heure
Price	"Stub" pour le changement de coordonnées prix
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier

© 2000-2015, MetaQuotes Software Corp.

virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Propriétés des objet, Angle du graphique, Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "BmpLabel".

```
bool  Create(
   long     chart_id,      // Identifiant du graphique
   string   name,          // Nom de l'objet
   int      window,        // Fenêtre du graphique
   int      X,             // Coordonnée X
   int      Y              // Coordonnée Y
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

X

[in] Coordonnée X.

Y

[in] Coordonnée Y.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

X_Distance (Méthode "Get")

Retourne la valeur de la propriété "X_Distance".

```
int  X_Distance() const
```

Valeur de retour

Valeur de la propriété "X_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

X_Distance (Méthode "Set")

Définit une nouvelle valeur pour la propriété "X_Distance".

```
bool  X_Distance(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur pour la propriété "X_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Y_Distance (Méthode "Get")

Retourne la valeur de la propriété "Y_Distance".

```
int  Y_Distance() const
```

Valeur de retour

Valeur de la propriété "Y_Distance" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

Y_Distance (Méthode "Set")

Définit la nouvelle valeur de la propriété "Y_Distance".

```
bool  Y_Distance(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur pour la propriété "Y_Distance".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

X_Offset (Méthode "Get")

Retourne la valeur de la propriété "X_Offset" (la coordonnée X du coin supérieur gauche du rectangle visible de l'objet).

```
int  X_Offset() const
```

Valeur de retour

Valeur de la propriété "X_Offset" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

X_Offset (Méthode "Set")

Définit la nouvelle valeur de la propriété "X_Offset" (la coordonnée X du coin supérieur gauche du rectangle visible de l'objet). La valeur est définie en pixels relativement au coin suéprieur gauche de l'image originale.

```
bool  X_Offset(
   int  X      // nouvelle valeur
   )
```

Paramètres

X

[in] Nouvelle valeur de la propriété "X_Offset".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Y_Offset (Méthode "Get")

Retourne la valeur de la propriété "Y_Offset" (la coordonnée Y du coin supérieur gauche du rectangle visible de l'objet).

```
int  Y_Offset() const
```

Valeur de retour

Valeur de la propriété "Y_Offset" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

Y_Offset (Méthode "Set")

Définit la nouvelle valeur de la propriété "Y_Offset" (la coordonnée Y du coin supérieur gauche du rectangle visible de l'objet). La valeur est définie en pixels relativement au coin suéprieur gauche de l'image originale.

```
bool  Y_Offset(
   int  Y      // nouvelle valeur
   )
```

Paramètres

Y

[in] Nouvelle valeur de la propriété "Y_Offset".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Corner (Méthode "Get")

Retourne la valeur de la propriété "Corner".

```
ENUM_BASE_CORNER Corner() const
```

Valeur de retour

Valeur de la propriété "Corner" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Corner (Méthode "Set")

Définit une nouvelle pour la propriété "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // nouvelle valeur
   )
```

Paramètres

corner

[in] Nouvelle valeur pour la propriété "Corner".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

X_Size

Retourne la valeur de la propriété "X_Size".

```
int  X_Size() const
```

Valeur de retour

Valeur de la propriété "X_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

Y_Size

Retourne la valeur de la propriété "Y_Size".

```
int  Y_Size() const
```

Valeur de retour

Valeur de la propriété "Y_Size" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

BmpFileOn (Méthode "Get")

Retourne la valeur de la propriété "BmpFileOn".

```
string  BmpFileOn() const
```

Valeur de retour

Valeur de la propriété "BmpFileOn" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne "".

BmpFileOn (Méthode "Set")

Définit une nouvelle valeur pour la propriété "BmpFileOn".

```
bool  BmpFileOn(
   string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nouvelle valeur pour la propriété "BmpFileOn".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

BmpFileOff (Méthode "Get")

Retourne la valeur de la propriété "BmpFileOff".

```
string  BmpFileOff() const
```

Valeur de retour

Valeur de la propriété "BmpFileOff" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne "".

BmpFileOff (Méthode "Set")

Définit une nouvelle valeur pour la propriété "BmpFileOff".

```
bool  BmpFileOff(
   string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nouvelle valeur pour la propriété "BmpFileOff".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

State (Méthode "Get")

Retourne la valeur de la propriété "State".

```
bool  State() const
```

Valeur de retour

Valeur de la propriété "State" de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne faux.

State (Méthode "Set")

Définit une nouvelle valeur pour la propriété "State".

```
bool  State(
   bool  state      // nouvelle valeur d'état
   )
```

Paramètres

state

[in] Nouvelle valeur pour la propriété "State".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Time

Interdit le changement de la coordonnée date/heure.

```
bool  Time(
   datetime  time      // n'importe quelle valeur
   )
```

Paramètres

time

[in] N'importe quelle valeur de type datetime.

Valeur de retour

toujours faux.

© 2000-2015, MetaQuotes Software Corp.

Price

Interdit le changement de la coordonnée prix.

```
bool  Price(
   double  price      // n'importe quelle valeur
   )
```

Paramètres

price

[in] N'importe quelle valeur de type double.

Valeur de retour

toujours faux.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_BITMAP_LABEL pour CChartObjectBmpLabel).

© 2000-2015, MetaQuotes Software Corp.

CChartObjectRectLabel

La classe CChartObjectRectLabel permet un accès simplifié aux propriétés de l'objet graphique "Label Rectangle".

Description

La classe CChartObjectRectLabel fournit un accès aux propriétés de l'objet "Label Rectangle".

Déclaration

```
class CChartObjectRectLabel : public CChartObjectLabel
```

Titre

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Méthodes de Classe

Create	
Create	Crée un objet graphique "Label Rectangle"
Propriétés	
X_Size	Définit la taille horizontale
Y_Size	Définit la taille verticale
BackColor	Retourne/Définit la couleur d'arrière plan
Angle	Une méthode générée par un stub
BorderType	Retourne/Définit le type de bordure
Entrée/Sortie	
virtual Save	Méthode virtuelle pour écrire l'objet dans un fichier
virtual Load	Méthode virtuelle pour charger l'objet depuis un fichier
virtual Type	Méthode virtuelle d'identification

Voir aussi

Types des objets, Propriétés de l'objet, Objet graphiques

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un objet graphique de type "CChartObjectRectLabel".

```
bool  Create(
   long     chart_id,      // Identifiant du graphique
   string   name,          // Nom de l'objet
   int      window,        // Fenpetre du graphique
   int      X,             // Coordonnée X
   int      Y,             // Coordonnée Y
   int      sizeX,         // Taille horizontale
   int      sizeY          // Taille verticale
   )
```

Paramètres

chart_id

[in] Identifiant du graphique (0 - graphique courant).

name

[in] Un nom unique de l'objet à créer.

window

[in] Numéro de la fenêtre du graphique (0 - fenêtre principale).

X

[in] Coordonnée X.

Y

[in] Coordonnée Y.

sizeX

[in] Taille horizontale.

sizeY

[in] Taille verticale.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

X_Size

Définit la valeur de la propriété "X_Size".

```
bool  X_Size(
   int  size      // Taille horizontale
   )
```

Paramètres

size

[in] Nouvelle taille horizontale.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

Note

Pour récupérer les valeurs des propriétés "X_Size" et "Y_Size", utiliser les méthodes X_Size et Y_Size de la classe parente CChartObjectLabel.

© 2000-2015, MetaQuotes Software Corp.

Y_Size

Définit la valeur de la propriété "Y_Size".

```
bool  Y_Size(
   int  size      // Taille verticale
   )
```

Paramètres

size

[in] Nouvelle taille verticale.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

Note

Pour récupérer les valeurs des propriétés "X_Size" et "Y_Size", utiliser les méthodes X_Size et Y_Size de la classe parente CChartObjectLabel.

© 2000-2015, MetaQuotes Software Corp.

BackColor

Retourne la couleur d'arrière plan.

```
color  BackColor() const
```

Valeur de retour

Couleur d'arrière plan de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

BackColor

Définit la couleur d'arrière plan.

```
bool  BackColor(
   color  new_color      // Nouvelle couleur
   )
```

Paramètres

new_color

[in] Nouvelle couleur d'arrière plan.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Angle

Une méthode générée vi aun stub.

```
bool  Angle(
   double  angle      // n'importe quelle valeur
   )
```

Paramètres

angle

[in] N'importe quelle valeur de type double type.

Valeur de retour

Toujours faux.

© 2000-2015, MetaQuotes Software Corp.

BorderType

Retourne le type de bordure.

```
int  BorderType() const
```

Valeur de retour

Type de bortdure de l'objet assigné à l'instance de la classe. Si aucun objet n'est assigné, retourne 0.

BorderType

Définit le type de bordure.

```
bool  BorderType(
   int  type      // Type de bordure
   )
```

Paramètres

type

[in] Nouveau type de bordure.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire précédemment ouvert avec la fonction FileOpen.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identificant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (OBJ_RECTANGLE_LABEL pour CChartObjectRectangleLabel).

© 2000-2015, MetaQuotes Software Corp.

CCanvas

La classe CCanvas permet la création simplifiée d'images.

Description

CCanvas fournit la création de ressources graphiques (avec ou sans attachement à l'objet graphique) et des primitives graphiques de dessin.

Déclaration

```
class CCanvas
```

Titre

```
#include <Canvas\Canvas.mqh>
```

Méthodes de Classe par Groupes

Création	
Create	Crée une ressource graphique sans le lier à un objet Graphique (chart)
CreateBitmap	Crée une ressource graphique liée à un objet Graphique (chart)
CreateBitmapLabel	Crée une ressource graphique liée à un objet Graphique (chart)
Destroy	Détruit une ressource graphique
Propriétés	
ChartObjectName	Retourne le nom d'un objet Graphique lié
ResourceName	Retourne le nom d'une ressource graphique
Width	Retourne la largeur d'une ressource graphique
Height	Retourne la hauteur d'une ressource graphique
LineStyleSet	Définit le style de ligne
Mise à jour d'un objet à l'écran	
Update	Affiche les changements à l'écran
Resize	Redimmensionne une ressource graphique
Supprimer/Remplir avec une couleur	
Erase	Supprime ou remplit avec la couleur spécifiée
Accès aux données	
PixelGet	Retourne la couleur du point situé aux coordonnées spécifiées

© 2000-2015, MetaQuotes Software Corp.

PixelSet	Définit la couleur du point situé aux coordonnées spécifiées
Primitives de dessin	
LineVertical	Dessine une ligne verticale
LineHorizontal	Dessine une ligne horizontale
Line	Dessine une ligne à main levée
Polyline	Dessine une polyline
Polygon	Dessine un polygone
Rectangle	Dessine un rectangle
Circle	Dessine un cercle
Triangle	Dessine un triangle
Primitives de dessin avec remplissage	
FillRectangle	Dessine une rectangle plein
FillCircle	Dessine un cercle plein
FillTriangle	Dessine un triangle plein
Fill	Remplit une zone
Primitives de dessin avec anti-aliasing	
PixelSetAA	Dessine un pixel
LineAA	Dessine une ligne
PolylineAA	Dessine une polyline
PolygonAA	Dessine un polygone
TriangleAA	Dessine un triangle
CircleAA	Dessine un cercle
Texte	
FontSet	Définit les paramètres de la police de caractères
FontNameSet	Définit le nom de la police de caractères
FontSizeSet	Retourne la taille de la police de caractères
FontFlagsSet	Définit les flags de la police de caractères
FontAngleSet	Définit l'angle de pente de la police de caractères
FontGet	Retourne les paramètres de la police de caractères
FontNameGet	Retourne le nom de la police de caractères

© 2000-2015, MetaQuotes Software Corp.

FontSizeGet	Retourne la taille de la police de caractères
FontFlagsGet	Retourne les flags de la police de caractères
FontAngleGet	Retourne l'angle de pente de la police de caractères
TextOut	Affiche un texte
TextWidth	Retourne la largeur du texte
TextHeight	Retourne la hauteur du texte
TextSize	Retourne la taille du texte
Transparence	
TransparentLevelSet	Définit le niveau de transparence
Entrée/Sortie	
LoadFromFile	Lit une image depuis un fichier BMP

© 2000-2015, MetaQuotes Software Corp.

ChartObjectName

Retourne le nom d'un objet lié à l'objet Graphique (chart).

```
string  ChartObjectName();
```

Valeur de retour

le nom d'un objet lié à l'objet graphique (chart)

© 2000-2015, MetaQuotes Software Corp.

Circle

Dessine un cercle

```
void  Circle(
   int          x,        // Coordonnée X
   int          y,        // Coordonnée Y
   int          r,        // rayon
   const uint   clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du centre du cercle.

y

[in] Coordonnée Y du centre du cercle.

r

[in] Rayon du cercle.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

CircleAA

Dessine un cercle en utilsant l'algorithme d'anti-aliasing

```
void  CircleAA(
   const int      x,        // coordonnée X
   const int      y,        // coordonnée Y
   const double   r,        // rayon
   const uint     clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du centre du cercle.

y

[in] Coordonnée Y du centre du cercle.

r

[in] Rayon du cercle.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée une ressource graphique sans la liér à l'objet Graphique (chart).

```
virtual bool  Create(
   const string       name,                              // nom
   const int          width,                             // largeur
   const int          height,                            // hauteur
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // format
   );
```

Paramètres

name

[in] Base du nom d'une ressource graphique. Un nom de ressource est généré à la création en ajoutant une chaîne pseudo-aléatoire.

width

[in] Largeur (taille selon l'axe X) en pixels.

height

[in] Hauteur (taille selon l'axe Y) en pixels.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Méthode de traitement de la couleur. Voir la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de traitement des couleurs.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateBitmap

Crée une ressource graphique liée à l'objet Graphique (chart).

1. Crée une ressource graphique dans la fenêtre principale du graphique actuel.

```
bool  CreateBitmap(
   const string       name,                                  // nom
   const datetime     time,                                  // date/heure
   const double       price,                                 // prix
   const int          width,                                 // largeur
   const int          height,                                // hauteur
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

2. Crée une ressource graphique en utilisant l'identifiant d'un graphique et un numéro de sous-fenêtre.

```
bool  CreateBitmap(
   const long         chart_id,                              // identifiant d'un graphi
   const int          subwin,                                // numéro de la sous-fenêt
   const string       name,                                  // nom
   const datetime     time,                                  // date/heure
   const double       price,                                 // prix
   const int          width,                                 // largeur
   const int          height,                                // hauteur
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

Paramètres

chart_id

[in] Identifiant d'un graphique pour créer un objet.

subwin

[in] Numéro de la sous-fenêtre d'un graphique pour créer un objet.

name

[in] Nom de l'objet Graphique (chart) et base de nom pour une ressource graphique.

time

[in] Coordonnée dans le temps du point d'ancrage de l'objet.

price

[in] Coordonnée en valeur du point d'ancrage de l'objet.

width

[in] Largeur de la ressource graphique (taille selon l'axe X) en pixels.

height

[in] Hauteur de la ressource graphique (taille selong l'axe Y) en pixels.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2015, MetaQuotes Software Corp.

[in] Méthode de traitement de la couleur. Voir la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de traitement des couleurs.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si la première fonction est utilisée, l'objet est créé dans la fenêtre principale du graphique courant.

La taille d'un objet correspond à la taille d'une ressource graphique.

© 2000-2015, MetaQuotes Software Corp.

CreateBitmapLabel

Crée une ressource graphique liée à l'objet Graphique (chart).

1. Crée une ressource graphique dans la fenêtre principale du graphique actuel.

```
bool  CreateBitmapLabel(
   const string       name,                              // nom
   const int          x,                                 // coordonnée X
   const int          y,                                 // coordonnée Y
   const int          width,                             // largeur
   const int          height,                            // hauteur
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // format
   );
```

2. Crée une ressource graphique en utilisant l'identifiant d'un graphique et un numéro de sous-fenêtre.

```
bool  CreateBitmapLabel(
   const long         chart_id,                          // identifiant d'un graphi
   const int          subwin,                            // numéro de la sous-fenêt
   const string       name,                              // nom
   const int          x,                                 // coordonnée X
   const int          y,                                 // coordonnée Y
   const int          width,                             // largeur
   const int          height,                            // hauteur
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // format
   );
```

Paramètres

chart_id

[in] Identifiant d'un graphique pour créer un objet.

subwin

[in] Numéro de la sous-fenêtre d'un graphique pour créer un objet.

name

[in] Nom de l'objet Graphique (chart) et base de nom pour une ressource graphique.

x

[in] Coordonnée X du point d'ancrage de l'objet Graphique.

y

[in] Coordonnée Y du point d'ancrage de l'objet Graphique.

width

[in] Largeur de la ressource graphique (taille selon l'axe X) en pixels.

height

[in] Hauteur de la ressource graphique (taille selong l'axe Y) en pixels.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2015, MetaQuotes Software Corp.

[in] Méthode de traitement de la couleur. Voir la description de la fonction ResourceCreate() pour en savoir plus sur les méthodes de traitement des couleurs.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si la première fonction est utilisée, l'objet est créé dans la fenêtre principale du graphique courant.

La taille d'un objet correspond à la taille d'une ressource graphique.

© 2000-2015, MetaQuotes Software Corp.

Destroy

Détruit une ressource graphique.

```
void  Destroy();
```

Note

Si une ressource graphique a été liée à un obhjet Graphique (chart), ce derner est détruit également.

© 2000-2015, MetaQuotes Software Corp.

Erase

Vide ou remplis avec la couleur spécifiée.

```
void  Erase(
   const uint  clr=0      // couleur
   );
```

Paramètres

clr=0

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

Fill

Remplis une zone.

```
void  Fill(
   int          x,        // Coordonnée X
   int          y,        // Coordonnée Y
   const uint   clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du point de départ du remplissage.

y

[in] Coordonnée Y du point de départ du remplissage.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

FillCircle

Dessine un cercle plein.

```
void  FillCircle(
   int          x,        // Coordonnée X
   int          y,        // Coordonnée Y
   int          r,        // rayon
   const uint   clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du centre du cercle plein.

y

[in] Coordonnée Y du centre du cercle plein.

r

[in] Rayon du cercle plein.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

FillRectangle

Dessine un rectangle plein.

```
void  FillRectangle(
   int          x1,      // coordonnée X
   int          y1,      // coordonnée Y
   int          x2,      // coordonnée X
   int          y2,      // coordonnée Y
   const uint   clr      // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier point formant le rectangle.

y1

[in] Coordonnée Y du premier point formant le rectangle.

x2

[in] Coordonnée X du second point formant le rectangle.

y2

[in] Coordonnée Y du second point formant le rectangle.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

FillTriangle

Dessine un triangle plein.

```
void  FillTriangle(
   int          x1,      // coordonnée X
   int          y1,      // coordonnée Y
   int          x2,      // coordonnée X
   int          y2,      // coordonnée Y
   int          x3,      // coordonnée X
   int          y3,      // coordonnée Y
   const uint   clr      // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier sommet du triangle.

y1

[in] Coordonnée Y du premier sommet du triangle.

x2

[in] Coordonnée X du second sommet du triangle.

y2

[in] Coordonnée Y du second sommet du triangle.

x3

[in] Coordonnée X du troisième sommet du triangle.

y3

[in] Coordonnée Y du troisième sommet du triangle.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

FontAngleGet

Retourne l'angle de pente de la police de caractères.

```
uint  FontAngleGet();
```

Valeur de retour

angle de pente de la police de caractères

© 2000-2015, MetaQuotes Software Corp.

FontAngleSet

Définis l'angle de pente de la police de caractères.

```
bool  FontAngleSet(
   uint  angle      // angle
   );
```

Paramètres

angle

[in] Angle de pente de la police de caractères en dixièmes de degrés.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

FontFlagsGet

Retourne les flags de la police de caractères.

```
uint  FontFlagsGet();
```

Valeur de retour

flags de la police de caractères

© 2000-2015, MetaQuotes Software Corp.

FontFlagsSet

Définis les flags de la police de caractères.

```
bool  FontFlagsSet(
   uint  flags      // flags
   );
```

Paramètres

flags

[in] Flags de créations de la police de caractères. Voir la description de la fonction TextSetFont() pour en savoir plus sur les flags.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

FontGet

Retourne les paramètres actuels de la police de caractères.

```
void  FontGet(
   string&  name,      // nom
   int&     size,      // taille
   uint&    flags,     // flags
   uint&    angle      // angle d'inclinaison
   );
```

Paramètres

name

[out] Référence à une variable pour retourner le nom de la police de caractères.

size

[out] Référence à une variable pour retourner la taille de la police de caractères.

flags

[out] Référence à une variable pour retourner les flags de la police de caractères.

angle

[out] Référence à une variable pour retourner l'angle de pente de la police de caractères.

© 2000-2015, MetaQuotes Software Corp.

FontNameGet

Retourne le nom de la police de caractères.

```
string  FontNameGet();
```

Valeur de retour

nom de la police de caractères

© 2000-2015, MetaQuotes Software Corp.

FontNameSet

Définit le nom de la police de caractères.

```
bool  FontNameSet(
   string  name      // nom
   );
```

Paramètres

name

[in] Nom de la police de caractères. Par exemple, "Arial".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

FontSet

Définit la police de caractères actuelle.

```
bool  FontSet(
   const string  name,        // nom
   const int     size,        // taille
   const uint    flags=0,     // flags
   const uint    angle=0      // angle
   );
```

Paramètres

name

[in] Nom de la police de caractères. Par exemple, "Arial".

size

[in] Taille de la police de caractères. Voir la description de la fonction TextSetFont() pour savoir comment définir la taille.

flags=0

[in] Flags de créations de la police de caractères. Voir la description de la fonction TextSetFont() pour en savoir plus sur les flags.

angle=0

[in] Angle de pente de la police de caractères en dixièmes de degrés.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

FontSizeGet

Retourne la taille de la police de caractères.

```
int  FontSizeGet();
```

Valeur de retour

taille de la police de caractères

© 2000-2015, MetaQuotes Software Corp.

FontSizeSet

Définit la taille de la police de caractères.

```
bool  FontSizeSet(
   int  size      // taille
   );
```

Paramètres

size

[in] Taille de la police de caractères. Voir la description de la fonction TextSetFont() pour savoir comment définir la taille.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Height

Retourne la taille d'une ressource graphique.

```
int  Height();
```

Valeur de retour

hauteur d'une ressource graphique

© 2000-2015, MetaQuotes Software Corp.

Line

Dessine un segment d'une ligne à main levée.

```
void  Line(
   int          x1,      // coordonnée X
   int          y1,      // coordonnée Y
   int          x2,      // coordonnée X
   int          y2,      // coordonnée Y
   const uint   clr      // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier point du segment.

y1

[in] Coordonnée y du premier point du segment.

x2

[in] Coordonnée X du second point du segment.

y2

[in] Coordonnée Y du second point du segment.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

LineAA

Dessine un segment d'une ligne à main levée en utilisant l'algorithme d'anti-aliasing.

```
void  LineAA(
   const int   x1,                  // coordonnée X
   const int   y1,                  // coordonnée Y
   const int   x2,                  // coordonnée X
   const int   y2,                  // coordonnée Y
   const uint  clr,                 // couleur
   const uint  style=UINT_MAX       // style de ligne
   );
```

Paramètres

x1

[in] Coordonnée X du premier point du segment.

y1

[in] Coordonnée y du premier point du segment.

x2

[in] Coordonnée X du second point du segment.

y2

[in] Coordonnée Y du second point du segment.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2015, MetaQuotes Software Corp.

LineHorizontal

Dessine un segment d'une ligne horizontale.

```
void  LineHorizontal(
   int          x1,       // coordonnée X
   int          x2,       // coordonnée X
   int          y,        // Coordonnée Y
   const uint   clr       // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du segment.

x2

[in] Coordonnée X du premier point du segment.

y

[in] Coordonnée Y du second point du segment.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

LineStyleSet

Définit le style de ligne.

```
void  LineStyleSet(
   const uint  style      // style
   );
```

Paramètres

style

[in] Style de ligne.

Note

Le paramètre d'entrée peut avoir n'importe quelle valeur de l'énumération ENUM_LINE_STYLE. Il est également possible de créer un style de ligne personnalisée.

© 2000-2015, MetaQuotes Software Corp.

LineVertical

Dessine un segment de ligne verticale.

```
void  LineVertical(
   int          x,        // Coordonnée X
   int          y1,       // coordonnée Y
   int          y2,       // coordonnée Y
   const uint   clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du segment.

y1

[in] Coordonnée y du premier point du segment.

y2

[in] Coordonnée Y du second point du segment.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

LoadFromFile

Lit une image depuis un fichier BMP.

```
bool  LoadFromFile(
   const string  filename      // nom du fichier
   );
```

Paramètres

filename

[in] Nom du fichier (incluant l'extension "BMP").

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

PixelGet

Retourne la couleur du point situé aux coordonnées spécifiées.

```
uint  PixelGet(
   const int  x,     // coordonnée X
   const int  y      // coordonnée Y
   );
```

Paramètres

x

[in] Coordonnée X du point.

y

[in] Coordonnée Y du point.

Valeur de retour

Couleur du point au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

PixelSet

Définit la couleur du point situé aux coordonnées spécifiées.

```
void  PixelSet(
   const int   x,       // coordonnée X
   const int   y,       // coordonnée Y
   const uint  clr      // couleur
   );
```

Paramètres

x

[in] Coordonnée X du point.

y

[in] Coordonnée Y du point.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

PixelSetAA

Dessine un point en utilisant l'algorithme d'anti-aliasing.

```
void  PixelSetAA(
   const double  x,        // coordonnée X
   const double  y,        // coordonnée Y
   const uint    clr       // couleur
   );
```

Paramètres

x

[in] Coordonnée X du point.

y

[in] Coordonnée Y du point.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

Polygon

Dessine un polygone.

```
void  Polygon(
   int&         x[],     // tableau de coordonnées X
   int&         y[],     // tableau de coordonnées Y
   const uint   clr      // couleur
   );
```

Paramètres

x[]

[in] Tableau de coordonnées X des points d'un polygone.

y[]

[in] Tableau de coordonnées Y des points d'un polygone.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

PolygonAA

Dessine un polygone en utilisant l'algorithme d'anti-aliasing.

```
void  PolygonAA(
   int&         x[],                // tableau de coordonnées X
   int&         y[],                // tableau de coordonnées Y
   const uint   clr,                // couleur
   const uint   style=UINT_MAX      // style de ligne
   );
```

Paramètres

x[]

[in] Tableau de coordonnées X des points d'un polygone.

y[]

[in] Tableau de coordonnées Y des points d'un polygone.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2015, MetaQuotes Software Corp.

Polyline

Dessine une polyline.

```
void  Polyline(
   int&         x[],     // tableau de coordonnées X
   int&         y[],     // tableau de coordonnées Y
   const uint   clr      // couleur
   );
```

Paramètres

x[]

[in] Tableau de coordonnées X de la polyline.

y[]

[in] Tableau de coordonnées Y de la polyline.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

PolylineAA

Dessine une polyline en utilisant l'algorithme d'anti-aliasing.

```
void  PolylineAA(
   int&         x[],                // tableau de coordonnées X
   int&         y[],                // tableau de coordonnées Y
   const uint   clr,                // couleur
   const uint   style=UINT_MAX      // style de ligne
   );
```

Paramètres

x[]

[in] Tableau de coordonnées X de la polyline.

y[]

[in] Tableau de coordonnées Y de la polyline.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2015, MetaQuotes Software Corp.

Rectangle

Dessine un rectangle à partir de 2 points.

```
void  Rectangle(
   int          x1,      // coordonnée X
   int          y1,      // coordonnée Y
   int          x2,      // coordonnée X
   int          y2,      // coordonnée Y
   const uint   clr      // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier point formant le rectangle.

y1

[in] Coordonnée Y du premier point formant le rectangle.

x2

[in] Coordonnée X du second point formant le rectangle.

y2

[in] Coordonnée Y du second point formant le rectangle.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

Resize

Redimmensionne une ressource graphique.

```
bool  Resize(
   const int  width,       // largeur
   const int  height       // hauteur
   );
```

Paramètres

width

[in] Nouvelle largeur de la ressource graphique.

height

[in] Nouvelle hauteur de la ressource graphique.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Lors du redimmensionnement, l'image précédente n'est pas sauvegardée.

© 2000-2015, MetaQuotes Software Corp.

ResourceName

Retourne le nom d'une ressource graphique.

```
string  ResourceName();
```

Valeur de retour

nom d'une ressource graphique

© 2000-2015, MetaQuotes Software Corp.

TextHeight

Retourne la hauteur du texte.

```
int  TextHeight(
   const string  text      // texte
   );
```

Paramètres

text

[in] Texte à mesurer.

Valeur de retour

hauteur du texte en pixels

Note

La police de caractères actuelle est utilisée pour mesurer le texte.

© 2000-2015, MetaQuotes Software Corp.

TextOut

Affiche le texte.

```
void  TextOut(
   int          x,               // coordonnée X
   int          y,               // coordonnée Y
   string       text,            // texte
   const uint   clr,             // couleur
   uint         alignment=0      // alignement
   );
```

Paramètres

x

[in] Coordonnée X du point d'ancrage du texte.

y

[in] Coordonnée Y du point d'ancrage du texte.

text

[in] Texte à afficher.

clr

[in] Couleur au format ARGB.

alignment=0

[in] Méthode d'ancrage du texte. Voir la description de la fonction TextOut() pour en savoir plus sur les différentes méthodes d'ancrage.

Note

La police de caractères actuelle est utilisée pour afficher le texte.

© 2000-2015, MetaQuotes Software Corp.

TextSize

Retourne la taille du texte.

```
void  TextSize(
   const string  text,        // texte
   int&          width,       // largeur
   int&          height       // hauteur
   );
```

Paramètres

text

[in] Texte à mesurer.

width

[out] Référence à une variable pour retourner la largeur du texte.

height

[out] Référence à une variable pour retourner la hauteur du texte.

Note

La police de caractères actuelle est utilisée pour mesurer le texte.

© 2000-2015, MetaQuotes Software Corp.

TextWidth

Retourne la largeur de texte.

```
int  TextWidth(
   const string  text      // texte
   );
```

Paramètres

text

[in] Texte à mesurer.

Valeur de retour

hauteur du texte en pixels

Note

La police de caractères actuelle est utilisée pour mesurer le texte.

© 2000-2015, MetaQuotes Software Corp.

TransparentLevelSet

Définit le niveau de transparence.

```
void  TransparentLevelSet(
   const uchar  value      // valeur
   );
```

Paramètres

value

[in] Nouvelle valeur du niveau de transparence.

Note

0 signifie transparence complète, tandis que 255 signifie opacité totale.

Définir le niveau de transparence affecte tous les objets précédemment dessinés. Le niveau de transparence spécifié n'affecte pas les créations suivantes.

© 2000-2015, MetaQuotes Software Corp.

Triangle

Dessine un triangle.

```
void  Triangle(
   int          x1,      // coordonnée X
   int          y1,      // coordonnée Y
   int          x2,      // coordonnée X
   int          y2,      // coordonnée Y
   int          x3,      // coordonnée X
   int          y3,      // coordonnée Y
   const uint   clr      // couleur
   );
```

Paramètres

x1

[in] Coordonnée X du premier sommet du triangle.

y1

[in] Coordonnée Y du premier sommet du triangle.

x2

[in] Coordonnée X du second sommet du triangle.

y2

[in] Coordonnée Y du second sommet du triangle.

x3

[in] Coordonnée X du troisième sommet du triangle.

y3

[in] Coordonnée Y du troisième sommet du triangle.

clr

[in] Couleur au format ARGB.

© 2000-2015, MetaQuotes Software Corp.

TriangleAA

Dessine un triangle en utilisant l'algorithme d'anti-aliasing.

```
void  TriangleAA(
   const int   x1,                  // coordonnée X
   const int   y1,                  // coordonnée Y
   const int   x2,                  // coordonnée X
   const int   y2,                  // coordonnée Y
   const int   x3,                  // coordonnée X
   const int   y3,                  // coordonnée Y
   const uint  clr,                 // couleur
   const uint  style=UINT_MAX       // style de ligne
   );
```

Paramètres

x1

[in] Coordonnée X du premier sommet du triangle.

y1

[in] Coordonnée Y du premier sommet du triangle.

x2

[in] Coordonnée X du second sommet du triangle.

y2

[in] Coordonnée Y du second sommet du triangle.

x3

[in] Coordonnée X du troisième sommet du triangle.

y3

[in] Coordonnée Y du troisième sommet du triangle.

clr

[in] Couleur au format ARGB.

style=UINT_MAX

[in] Le style de ligne est une valeur de l'énumération ENUM_LINE_STYLE ou une valeur personnalisée.

© 2000-2015, MetaQuotes Software Corp.

Update

Affiche les changements sur l'écran.

```
void  Update(
   const bool  redraw=true      // flag
   );
```

Paramètres

redraw=true

Flag d'un graphique pour demander le rafraîchissement.

© 2000-2015, MetaQuotes Software Corp.

Width

Retourne la largeur d'une ressource graphique.

```
int  Width();
```

Valeur de retour

largeur de la ressource graphique

© 2000-2015, MetaQuotes Software Corp.

CChart

La classe CChart est une classe permettant un accès simplifié aux propriétés de l'objet graphique "Chart".

Description

La classe CChart fournit un accès aux propriétés de l'objet "Chart".

Déclaration

```
class CChart : public CObject
```

Titre

```
#include <Charts\Chart.mqh>
```

Méthodes de Classe

Accès aux données protégées	
ChartID	Retourne l'identifiant du graphique
Propriétés générales	
Mode	Définit/Retourne la valeur de la propriété "Mode" (barre, bougie ou ligne)
Foreground	Définit/Retourne la valeur de la propriété "Foreground"
Shift	Définit/Retourne la valeur de la propriété "Shift"
ShiftSize	Définit/Retourne la valeur de la propriété "ShiftSize" (en pourcentage)
AutoScroll	Définit/Retourne la valeur de la propriété "AutoScroll"
Scale	Définit/Retourne la valeur de la propriété "Scale"
ScaleFix	Définit/Retourne la valeur de la propriété "ScaleFix" (échelle de graphique fixe ou pas)
ScaleFix_11	Définit/Retourne la valeur de la propriété "ScaleFix_11" (échelle de graphique 1:1, ou pas)
FixedMax	Définit/Retourne la valeur de la propriété "FixedMax" (prix maximum fixé)
FixedMin	Définit/Retourne la valeur de la propriété "FixedMin" (prix minimal fixé)
ScalePPB	Définit/Retourne la valeur de la propriété

© 2000-2015, MetaQuotes Software Corp.

	"ScalePPB" (l'échelle est "point par barre" ou pas)
PointsPerBar	Définit/Retourne la valeur de la propriété "PointsPerBar" (en points par barre)
Afficher les propriétés	
ShowOHLC	Définit/Retourne la valeur de la propriété "ShowOHLC"
ShowLineBid	Définit/Retourne la valeur de la propriété "ShowLineBid"
ShowLineAsk	Définit/Retourne la valeur de la propriété "ShowLineAsk"
ShowLastLine	Définit/Retourne la valeur de la propriété "ShowLastLine"
ShowPeriodSep	Définit/Retourne la valeur de la propriété "ShowPeriodSep" (afficher les séparateurs de période)
ShowGrid	Définit/Retourne la valeur de la propriété "ShowGrid"
ShowVolumes	Définit/Retourne la valeur de la propriété "ShowVolumes" (couleur des volumes et des niveaux des positions ouvertes)
ShowObjectDescr	Définit/Retourne la valeur de la propriété "ShowObjectDescr" (afficher la description des objets graphiques)
ShowDateScale	Définit/Retourne la valeur de la propriété "ShowDateScale" (échelle de date du graphique)
ShowPriceScale	Définit/Retourne la valeur de la propriété "ShowPriceScale" (échelle des prix du graphique)
Couleurs	
ColorBackground	Définit/Retourne la valeur de la propriété "ColorBackground" (couleur d'arrière plan du graphique)
ColorForeground	Définit/Retourne la valeur de la propriété "ColorForeground" (couleur des axes, de l'échelle et des textes OHLC)
ColorGrid	Définit/Retourne la valeur de la propriété "ColorGrid" (couleur de la grille)
ColorBarUp	Définit/Retourne la valeur de la propriété "ColorBarUp" (couleur des barres haussières, leurs ombres et le contour des corps des

© 2000-2015, MetaQuotes Software Corp.

	bougies)
ColorBarDown	Définit/Retourne la valeur de la propriété "ColorBarDown" (couleur des barres baissières, leurs ombres et le contour des corps des bougies)
ColorCandleBull	Définit/Retourne la valeur de la propriété "ColorCandleBull" (couleur du corps des bougies haussières)
ColorCandleBear	Définit/Retourne la valeur de la propriété "ColorCandleBear" (couleur du corps des bougies baissières)
ColorChartLine	Définit/Retourne la valeur de la propriété "ColorChartLine" (couleur des graphiques en ligne et des bougies en Doji)
ColorVolumes	Définit/Retourne la valeur de la propriété "ColorVolumes" (couleur des volumes et des niveaux des positions ouvertes)
ColorLineBid	Définit/Retourne la valeur de la propriété "ColorLineBid" (couleur de la ligne Bid)
ColorLineAsk	Définit/Retourne la valeur de la propriété "ColorLineAsk" (couleur de la ligne Ask)
ColorLineLast	Définit/Retourne la valeur de la propriété "ColorLineLast" (couleur de la ligne du dernier deal)
ColorStopLevels	Définit/Retourne la valeur de la propriété "ColorStopLevels" (couleur des niveaux de SL et TP)
Propriétés en lecture seule	
VisibleBars	Retourne le nombre total de barres visibles sur le graphique
WindowsTotal	Retourne le nombre total de fenêtres de graphiques, incluant les sous-fenêtres des indicateurs
WindowIsVisible	Retourne le flag de visibilité de la sous-fenêtre spécifiée du graphique
WindowHandle	Retourne le handler de la fenêtre du graphique (HWND)
FirstVisibleBar	Retourne le numéro de la première barre visible du graphique
WidthInBars	Retourne la largeur de la fenêtre en nombre de barres.

© 2000-2015, MetaQuotes Software Corp.

WidthInPixels	Retourne la largeur de la sous-fenêtre en pixels.
HeightInPixels	Retourne la hauteur de la sous-fenêtre en pixels.
PriceMin	Retourne le prix minimum de la sous-fenêtre spécifiée
PriceMax	Retourne le prix maximum de la sous-fenêtre spécifiée
Propriétés	
Attach	Assigne le graphique courant à l'instance de classe
FirstChart	Assigne le premier graphique du terminal client à l'instance de classe
NextChart	Assigne le graphique suivant du terminal client à l'instance de classe
Open	Ouvre un graphique avec les paramètres spécifiés et l'assigne à l'instance de classe
Detach	Détache le graphique de l'instance de classe
Close	Ferme le graphique assigné à l'instance de classe
BringToTop	Affiche le graphique par-dessus tous les autres graphiques
EventObjectCreate	Définit un flag pour envoyer les notifications d'évènements de création d'un nouvel objet à un graphique
EventObjectDelete	Définit un flag pour envoyer les notificiations d'évènements de suppression d'un objet à un graphique
Indicateurs	
IndicatorAdd	Ajoute un indicateur avec le handler spécifié dans la sous-fenêtre spécifiée du graphique
IndicatorDelete	Supprime un indicateur ayant le nom spécifié de la sous-fenêtre spécifiée du graphique
IndicatorsTotal	Retourne le nombre total d'indicateurs appliqués dans la sous-fenêtre spécifiée du graphique
IndicatorName	Retourne le nom court de l'indicateur de la sous-fenêtre spécifiée du graphique
Navigation	
Navigate	Se déplace dans le graphique

© 2000-2015, MetaQuotes Software Corp.

Accès à l'API MQL5	
Symbol	Retourne le symbole affiché dans le graphique
Period	Retourne l'espace de temps du graphique
Redraw	Redessine le graphique assigné à l'instance de classe
GetInteger	Retourne la valeur de la propriété correspondante de l'objet
SetInteger	Définit la nouvelle valeur de la propriété de type integer
GetDouble	Retourne la valeur de la propriété correspondante de l'objet
SetDouble	Définit la nouvelle valeur de la propriété de type double
GetString	Retourne la valeur de la propriété correspondante de l'objet
SetString	Définit la nouvelle valeur de la propriété de type string
SetSymbolPeriod	Change le symbole et l'espace de temps du graphique assigné à l'instance de classe
ApplyTemplate	Applique le modèle spécifié au graphique
ScreenShot	Crée une copie d'écran du graphique spécifié et le sauvegarde dans un fichier .gif
WindowOnDropped	Retourne le numéro de la sous-fenêtre correspondant au point de dépose de l'objet (expert ou script)
PriceOnDropped	Retourne le prix correspondant au point de dépose de l'objet (expert ou script)
TimeOnDropped	Retourne la date/heure correspondant au point de dépose de l'objet (expert ou script)
XOnDropped	Retourne la coordonnée X correspondant au point de dépose de l'objet (expert ou script)
YOnDropped	Retourne la coordonnée Y correspondant au point de dépose de l'objet (expert ou script)
Entrée/Sortie	
virtual Save	Sauvegarde les paramètres de l'objet dans un fichier
virtual Load	Charges les paramètres de l'objet depuis un fichier

© 2000-2015, MetaQuotes Software Corp.

virtual Type	Retourne l'identifiant du type de l'objet graphique

© 2000-2015, MetaQuotes Software Corp.

ChartID

Retourne l'identifiant du graphique.

```
long  ChartID() const
```

Valeur de retour

L'identifiant du graphique assigné à l'instance de classe. Si aucun objet n'est assigné, retourne -1.

© 2000-2015, MetaQuotes Software Corp.

Mode (Méthode "Get")

Retourne la valeur de la propriété "Mode" (barre, bougie ou ligne)

```
ENUM_CHART_MODE  Mode() const
```

Valeur de retour

Valeur de la propriété "Mode" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne WRONG_VALUE.

Mode (Méthode "Set")

Définit la valeur de la propriété "Mode" (barre, bougie ou ligne)

```
bool  Mode(
   ENUM_CHART_MODE  mode       // nouveau mode du graphique
   )
```

Paramètres

mode

[in] Mode du graphique (bougie, barre ou ligne) de l'énumération ENUM_CHART_MODE.

Valeur de retour

vrai si réalisé avec succès, faux si le mode n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Foreground (Méthode "Get")

Retourne la valeur de la propriété "Foreground".

```
bool  Foreground() const
```

Valeur de retour

Valeur de la propriété "Foreground" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Foreground (Méthode "Set")

Définit une nouvelle valeur à la propriété "Foreground".

```
bool  Foreground(
   bool  foreground      // nouvelle valeur
   )
```

Paramètres

foreground

[in] Nouvelle valeur de la propriété "Foreground".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Shift (Méthode "Get")

Retourne la valeur de la propriété "Shift".

```
bool  Shift() const
```

Valeur de retour

Valeur de la propriété "Shift" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

Shift (Méthode "Set")

Définit une nouvelle valeur à la propriété "Shift".

```
bool  Shift(
   bool  shift      // nouvelle valeur du flag
   )
```

Paramètres

shift

[in] Nouvelle valeur pour la propriété "Shift".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ShiftSize (Méthode "Get")

Retourne la valeur de la propriété "ShiftSize" (en pourcentage).

```
double  ShiftSize() const
```

Valeur de retour

Valeur de la propriété "ShiftSize" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

ShiftSize (Méthode "Set")

Définit la valeur de la propriété "ShiftSize" (en pourcentage)

```
bool  ShiftSize(
   double  shift_size      // nouvelle valeur de la propriété
   )
```

Paramètres

shift_size

[in] Nouvelle valeur de la propriété "ShiftSize" (en pourcentage).

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

AutoScroll (Méthode "Get")

Retourne la valeur de la propriété "AutoScroll".

```
bool  AutoScroll() const
```

Valeur de retour

Valeur de la propriété "AutoScroll" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

AutoScroll (Méthode "Set")

Définit une nouvelle valeur à la propriété "AutoScroll".

```
bool  AutoScroll(
   bool  autoscroll      // nouvelle valeur du flag
   )
```

Paramètres

autoscroll

[in] Nouvelle valeur pour la propriété "Autoscroll".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

Scale (Méthode "Get")

Retourne la valeur de la propriété "Scale".

```
int  Scale() const
```

Valeur de retour

Valeur de la propriété "Scale" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne 0.

Scale (Méthode "Set")

Définit une nouvelle valeur à la propriété "Scale".

```
bool  Scale(
   int  scale      // nouvelle valeur
   )
```

Paramètres

scale

[in] Nouvelle valeur pour la propriété "Scale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ScaleFix (Méthode "Get")

Retourne la valeur de la propriété "ScaleFix" (échelle de graphique fixe ou pas)

```
bool  ScaleFix() const
```

Valeur de retour

Valeur de la propriété "ScaleFix" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

ScaleFix (Méthode "Set")

Définit une nouvelle valeur à la propriété "ScaleFix".

```
bool  ScaleFix(
   bool  scale_fix      // nouvelle valeur
   )
```

Paramètres

scale_fix

[in] Nouvelle valeur pour la propriété "ScaleFix".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ScaleFix_11 (Méthode "Get")

Retourne la valeur de la propriété "ScaleFix_11" (échelle de graphique 1:1, ou autre)

```
bool  ScaleFix_11() const
```

Valeur de retour

Valeur de la propriété "ScaleFix_11" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

ScaleFix_11 (Méthode "Set")

Définit une nouvelle valeur à la propriété "ScaleFix_11".

```
bool  ScaleFix_11(
   string  scale_11      // nouvelle valeur
   )
```

Paramètres

scale_11

[in] Nouvelle valeur pour la propriété "ScaleFix_11".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

FixedMax (Méthode "Get")

Retourne la valeur de la propriété "FixedMax" (prix maximum fixé)

```
double  FixedMax() const
```

Valeur de retour

Valeur de la propriété "FixedMax" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

FixedMax (Méthode "Set")

Définit une nouvelle valeur à la propriété "FixedMax".

```
bool  FixedMax(
   double  max       // nouveau maximum fixé
   )
```

Paramètres

max

[in] Nouvelle valeur pour la propriété "FixedMax".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

FixedMin (Méthode "Get")

Retourne la valeur de la propriété "FixedMin" (prix minimal fixé)

```
double  FixedMin() const
```

Valeur de retour

Valeur de la propriété "Fixed" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

FixedMin (Méthode "Set")

Définit une nouvelle valeur à la propriété "FixedMin".

```
bool  FixedMax(
   double  min      // nouveau minimum fixé
   )
```

Paramètres

max

[in] Nouvelle valeur pour la propriété "FixedMin".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

PointsPerBar (Méthode "Get")

Retourne la valeur de la propriété "PointsPerBar" (en points par barre)

```
double  PointsPerBar() const
```

Valeur de retour

Valeur de la propriété "PointsPerBar" de l'objetassigné à l'instance de classe. Si aucun objet n'est assigné, retourne EMPTY_VALUE.

PointsPerBar (Méthode "Set")

Définit une nouvelle valeur à la propriété "PointsPerBar".

```
bool  PointsPerBar(
   double  ppb      // nouvelle échelle (en points par barre)
   )
```

Paramètres

ppb

[in] Nouvelle valeur d'échelle (en points par barre).

Valeur de retour

vrai si réalisé avec succès, faux si l'échelle n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ScalePPB (Méthode "Get")

Retourne la valeur de la propriété "ScalePPB" (l'échelle est "point par barre" ou autre)

```
bool  ScalePPB() const
```

Valeur de retour

Valeur de la propriété "ScalePPB" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

ScalePPB (Méthode "Set")

Définit une nouvelle valeur à la propriété "ScalePPB".

```
bool  ScalePPB(
   bool  scale_ppb      // nouvelle valeur du flag
   )
```

Paramètres

scale_ppb

[in] Nouvelle valeur pour la propriété "ScalePPB".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ShowOHLC (Méthode "Get")

Retourne la valeur de la propriété "ShowOHLC".

```
bool  ShowOHLC() const
```

Valeur de retour

Valeur de la propriété "ShowOHLC" de l'objet assigné à l'instance de classe. Si aucun objet n'est assigné, retourne faux.

ShowOHLC (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowOHLC".

```
bool  ShowOHLC(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowOHLC".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ShowLineBid (Méthode "Get")

Retourne la valeur de la propriété "ShowLineBid."

```
bool  ShowLineBid() const
```

Valeur de retour

Valeur de la propriété "ShowLineBid" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowLineBid (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowLineBid".

```
bool  ShowLineBid(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowLineBid".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ShowLineAsk (Méthode "Get")

Retourne la valeur de la propriété "ShowLineAsk".

```
bool  ShowLineAsk() const
```

Valeur de retour

Valeur de la propriété "ShowLineAsk" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowLineAsk (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowLineAsk".

```
bool  ShowLineAsk(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowLineAsk".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ShowLastLine (Méthode "Get")

Retourne la valeur de la propriété "ShowLastLine".

```
bool  ShowLastLine() const
```

Valeur de retour

Valeur de la propriété "ShowLastLine" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowLastLine (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowLastLine".

```
bool  ShowLastLine(
   bool  show      // nouvelle valeur du flag
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowLastLine".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ShowPeriodSep (Méthode "Get")

Retourne la valeur de la propriété "ShowPeriodSep" (afficher les séparateurs de période).

```
bool  ShowPeriodSep() const
```

Valeur de retour

Valeur de la propriété "ShowPeriodSep" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowPeriodSep (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowPeriodSep".

```
bool  ShowPeriodSep(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowPeriodSep".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ShowGrid (Méthode "Get")

Retourne la valeur de la propriété "ShowGrid".

```
bool  ShowGrid() const
```

Valeur de retour

Valeur de la propriété "ShowGrid" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowGrid (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowGrid".

```
bool  ShowGrid(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowGrid".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ShowVolumes (Méthode "Get")

Retourne la valeur de la propriété "ShowVolumes".

```
bool  ShowVolumes() const
```

Valeur de retour

Valeur de la propriété "ShowVolumes" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowVolumes (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowVolumes".

```
bool  ShowVolumes(
   bool  show      // nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowVolumes".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ShowObjectDescr (Méthode "Get")

Retourne la valeur de la propriété "ShowObjectDescr" (afficher la description des objets graphiques)

```
bool  ShowObjectDescr() const
```

Valeur de retour

Valeur de la propriété "ShowObjectDescr" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne faux.

ShowObjectDescr (Méthode "Set")

Définit une nouvelle valeur à la propriété "ShowObjectDescr".

```
bool  ShowObjectDescr(
   bool  show      // Nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowObjectDescr".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ShowDateScale

Définit une nouvelle valeur à la propriété "ShowDateScale".

```
bool  ShowDateScale(
   bool  show      // Nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowDateScale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ShowPriceScale

Définit une nouvelle valeur à la propriété "ShowPriceScale".

```
bool  ShowPriceScale(
   bool  show      // Nouvelle valeur
   )
```

Paramètres

show

[in] Nouvelle valeur pour la propriété "ShowPriceScale".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorBackground (Méthode "Get")

Retourne la valeur de la propriété "ColorBackground" (couleur d'arrière plan du graphique).

```
color  ColorBackground() const
```

Valeur de retour

Valeur de la propriété "ColorBackground" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorBackground (Méthode "Set")

Définit la nouvelle valeur de la propriété "ColorBackground".

```
bool  ColorBackground(
   color  new_color      // nouvelle couleur d'arrière plan
   )
```

Paramètres

new_color

[in] Nouvelle couleur d'arrière plan.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorForeground (Méthode "Get")

Retourne la valeur de la propriété "ColorForeground" (couleur de premier plan des axes, de l'échelle et des textes OHLC)

```
color  ColorForeground() const
```

Valeur de retour

Valeur de la propriété "ColorForeground" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorForeground (Méthode "Set")

Définit la valeur de la propriété "ColorForeground" (our les axes, l'échelle et les textes OHLC)

```
bool  ColorForeground(
   color  new_color      // Nouvelle couleur
   )
```

Paramètres

new_color

[in] Nouvelle couleur des axes, de l'échelle et des textes OHLC.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorGrid (Méthode "Get")

Retourne la valeur de la propriété "ColorGrid" (couleur de la grille).

```
color  ColorGrid() const
```

Valeur de retour

Valeur de la propriété "ColorGrid" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorGrid (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorGrid".

```
bool  ColorGrid(
   color  new_color      // nouvelle couleur de grille
   )
```

Paramètres

new_color

[in] Nouvelle couleur de grille.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorBarUp (Méthode "Get")

Retourne la valeur de la propriété "ColorBarUp" (couleur des barres haussières, leur ombre et le contour des corps des bougies).

```
color  ColorBarUp() const
```

Valeur de retour

Valeur de la propriété "ColorBarUp" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorBarUp (Méthode "Set")

Définit la nouvelle valeur de la propriété "ColorBarUp".

```
bool  ColorBarUp(
   color  new_color      // nouvelle couleur des barres haussières
   )
```

Paramètres

new_color

[in] Nouvelle couleur des barres haussières.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorBarDown (Méthode "Get")

Retourne la valeur de la propriété "ColorBarDown" (couleur des barres baissières, leur ombre et le contour du corps des bougies).

```
color  ColorBarDown() const
```

Valeur de retour

Valeur de la propriété "ColorBarDown" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorBarDown (Méthode "Set")

Définit la nouvelle valeur de la propriété "ColorBarDown".

```
bool  ColorBarDown(
   color  new_color      // nouvelle couleur des barres baissières
   )
```

Paramètres

new_color

[in] Nouvelle couleur des barres baissières.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorCandleBull (Méthode "Get")

Retourne la valeur de la propriété "ColorCandleBull" (couleur du corps des bougies haussières).

```
color  ColorCandleBull() const
```

Valeur de retour

Valeur de la propriété "ColorCandleBull" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorCandleBull (Méthode "Set")

Définit une nouvelle valeur pour la propriété "ColorCandleBull".

```
bool  ColorCandleBull(
   color  new_color      // nouvelle couleur du corps des bougies haussières
   )
```

Paramètres

new_color

[in] Nouvelle couleur du corps des bougies haussières.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorCandleBear (Méthode "Get")

Retourne la valeur de la propriété "ColorCandleBear" (couleur du corps des bougies baissières).

```
color  ColorCandleBear() const
```

Valeur de retour

Valeur de la propriété "ColorCandleBear" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorCandleBear (Méthode "Set")

Définit une nouvelle valeur pour la propriété "ColorCandleBear".

```
bool  ColorCandleBear(
   color  new_color      // nouvelle couleur du corps des bougies baissières
   )
```

Paramètres

new_color

[in] Nouvelle couleur du corps des bougies baissières.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorChartLine (Méthode "Get")

Retourne la valeur de la propriété "ColorChartLine" (couleur des graphiques en ligne et des bougies en Doji)

```
color  ColorChartLine() const
```

Valeur de retour

Valeur de la propriété "ColorChartLine" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorChartLine (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorChartLine".

```
bool  ColorChartLine(
   color  new_color      // nouvelle couleur des graphiques en ligne
   )
```

Paramètres

new_color

[in] Nouvelle couleur des graphiques en ligne (et des bougies en Doji).

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorVolumes (Méthode "Get")

Retourne la valeur de la propriété "ColorVolumes" (couleur des volumes et des niveaux des positions ouvertes)

```
color  ColorVolumes() const
```

Valeur de retour

Valeur de la propriété "ColorVolumes" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorVolumes (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorVolumes".

```
bool  ColorVolumes(
   color  new_color      // nouvelle couleur des volumes (niveaux des positions ouvert
   )
```

Paramètres

new_color

[in] Nouvelle couleur des volumes (niveaux des positions ouvertes).

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorLineBid (Méthode "Get")

Retourne la valeur de la propriété "ColorLineBid" (couleur de la ligne Bid)

```
color  ColorLineBid() const
```

Valeur de retour

Valeur de la propriété "ColorLineBid" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorLineBid (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorLineBid".

```
bool  ColorLineBid(
   color  new_color      // nouvelle couleur de la ligne Bid
   )
```

Paramètres

new_color

[in] Nouvelle couleur de la ligne Bid.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorLineAsk (Méthode "Get")

Retourne la valeur de la propriété "ColorLineAsk" (couleur de la ligne Ask)

```
color  ColorLineAsk() const
```

Valeur de retour

Valeur de la propriété "ColorLineAsk" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorLineAsk (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorLineAsk".

```
bool  ColorLineAsk(
   color  new_color      // nouvelle couleur de la ligne Ask
   )
```

Paramètres

new_color

[in] Nouvelle couleur de la ligne Ask.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorLineLast (Méthode "Get")

Retourne la valeur de la propriété "ColorLineLast" (couleur de la ligne du dernier deal)

```
color  ColorLineLast() const
```

Valeur de retour

Valeur de la propriété "ColorLineLast" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorLineLast (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorLineLast".

```
bool  ColorLineLast(
   color  new_color      // nouvelle couleur de la ligne du dernier deal
   )
```

Paramètres

new_color

[in] Nouvelle couleur de la ligne du dernier deal.

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ColorStopLevels (Méthode "Get")

Retourne la valeur de la propriété "ColorStopLevels" (couleur des niveaux de SL et TP)

```
color  ColorStopLevels() const
```

Valeur de retour

Valeur de la propriété "ColorStopLevels" du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne CLR_NONE.

ColorStopLevels (Méthode "Set")

Définit une nouvelle valeur à la propriété "ColorStopLevels".

```
bool  ColorStopLevels(
   color  new_color      // nouvelle couleur des niveaux de SL et TP
   )
```

Paramètres

new_color

[in] nouvelle couleur des niveaux de Stop Loss et Prix du Take Profit

Valeur de retour

vrai si réalisé avec succès, faux si la couleur n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

VisibleBars

Retourne le nombre total de barres visibles sur le graphique.

```
int  VisibleBars() const
```

Valeur de retour

Retourne le nombre total de barres visibles sur le graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

WindowsTotal

Retourne le nombre total de fenêtres de graphiques, incluant les sous-fenêtres des indicateurs.

```
int  WindowsTotal() const
```

Valeur de retour

Nombre total de fenêtres, incluant les sous-fenêtres des indicateurs, assignées à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

WindowIsVisible

Retourne le flag de visibilité de la sous-fenêtre spécifiée du graphique.

```
bool  WindowIsVisible(
   int  num      // numéro de la sous-fenêtre
   ) const
```

Paramètres

num

[in] Numéro de la sous-fenêtre (0 signifie la fenêtre principale).

Valeur de retour

Retourne le flag de visibilité de la sous-fenêtre spécifiée du graphique assigné à l'instance du graphique. Si aucun graphique n'est assigné, retourne faux.

© 2000-2015, MetaQuotes Software Corp.

WindowHandle

Retourne le handler de la fenêtre du graphique (HWND).

```
int  WindowHandle() const
```

Valeur de retour

Handler de la fenêtre du graphique assigné à l'instance du graphique. Si aucun graphique n'est assigné, retourne INVALID_HANDLE.

© 2000-2015, MetaQuotes Software Corp.

FirstVisibleBar

Retourne le numéro de la première barre visible du graphique.

```
int  FirstVisibleBar() const
```

Valeur de retour

Numéro de la première barre visible du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne -1.

© 2000-2015, MetaQuotes Software Corp.

WidthInBars

Retourne la largeur de la fenêtre en nombre de barres.

```
int  WidthInBars() const
```

Valeur de retour

Largeur de la fenêtre en barres assignée à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

WidthInPixels

Retourne la largeur de la sous-fenêtre en pixels.

```
int  WidthInPixels() const
```

Valeur de retour

Largeur en pixels de la sous-fenêtre du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

HeightInPixels

Retourne la hauteur de la sous-fenêtre en pixels.

```
int  HeightInPixels(
   int  num      // numéro de la sous-fenêtre
   ) const
```

Paramètres

num

[in] Numéro de la sous-fenêtre (0 signifie la fenêtre principale).

Valeur de retour

La hauteur de la sous-fenêtre du graphique en pixels. Si aucun graphique n'est assigné, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

PriceMin

Retourne le prix minimum de la sous-fenêtre spécifiée.

```
double  PriceMin(
   int  num      // numéro de la sous-fenêtre
   ) const
```

Paramètres

num

[in] Numéro de la sous-fenêtre (0 signifie la fenêtre principale).

Valeur de retour

Prix minimal du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne EMPTY_VALUE.

© 2000-2015, MetaQuotes Software Corp.

PriceMax

Retourne le prix maximum de la sous-fenêtre spécifiée.

```
double  PriceMax(
   int  num      // numéro de la sous-fenêtre
   ) const
```

Paramètres

num

[in] Numéro de la sous-fenêtre (0 signifie la fenêtre principale).

Valeur de retour

Prix maximal du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne EMPTY_VALUE.

© 2000-2015, MetaQuotes Software Corp.

Attach

Assigne le graphique actuel à l'instance de classe.

```
void  Attach()
```

Attach

Assigne le graphique spécifié à l'instance de classe.

```
void  Attach(
   long  chart      // Identifiant du graphique
   )
```

Paramètres

chart

[in] Identifiant du graphique à assigner.

© 2000-2015, MetaQuotes Software Corp.

FirstChart

Assigne le premier graphique du terminal client à l'instance de classe

```
void  FirstChart()
```

© 2000-2015, MetaQuotes Software Corp.

NextChart

Assigne le graphique suivant du terminal client à l'instance de classe.

```
void  NextChart()
```

© 2000-2015, MetaQuotes Software Corp.

Open

Ouvre le graphique avec les paramètres spécifiés et l'assigne à l'instance de classe.

```
long  Open(
   const string      symbol_name,      // Nom du symbole
   ENUM_TIMEFRAMES   timeframe         // Période
   )
```

Paramètres

symbol_name

[in] Nom du symbole. NULL signifie le symbole du graphique courant (auquel l'expert est attaché).

timeframe

[in] Espace de temps du graphique (énumération ENUM_TIMEFRAMES). 0 signifie l'espace de temps actuel.

Valeur de retour

Identifiant du graphique.

© 2000-2015, MetaQuotes Software Corp.

Detach

Détache le graphique de l'instance de classe

```
void  Detach()
```

© 2000-2015, MetaQuotes Software Corp.

Close

Ferme le graphique assigné à l'instance de classe.

```
void  Close()
```

© 2000-2015, MetaQuotes Software Corp.

BringToTop

Affiche le graphique par-dessus tous les autres graphiques.

```
bool  BringToTop() const
```

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

EventObjectCreate

Définit un flag pour envoyer les notifications d'un évènement de création d'un nouvel objet à tous les programmes MQL5 d'un graphique.

```
bool  EventObjectCreate(
   bool  flag      // flag
   )
```

Paramètres

flag

[in] Nouvelle valeur du flag.

Valeur de retour

vrai si réalisé avec succès, faux si le flag n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

EventObjectDelete

Définit un flag pour envoyer les notifications d'un évènement de suppression d'un objet à tous les programmes MQL5 d'un graphique.

```
bool  EventObjectDelete(
   bool  flag      // flag
   )
```

Paramètres

flag

[in] Nouvelle valeur du flag.

Valeur de retour

vrai si réalisé avec succès, faux si le flag n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

IndicatorAdd

Ajoute un indicateur avec le handler spécifié dans la sous-fenêtre spécifiée du graphique

```
bool  IndicatorAdd(
   int   sub_win         // numéro de la sous-fenêtre
   int   handle          // handle de l'indicateur
   );
```

Paramètres

sub_win

[in] Numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre du graphique principal. Si le numéro de sous-fenêtre spécifié n'existe pas, une nouvelle fenêtre est créée.

handle

[in] Handle de l'indicateur.

Valeur de Retour

La fonction retourne vrai en cas de succès, sinon retourne faux. Pour obtenir plus d'information concernant l'error, appeler la fonction GetLastError().

Voir aussi

IndicatorDelete(), IndicatorsTotal(), IndicatorName().

© 2000-2015, MetaQuotes Software Corp.

IndicatorDelete

Supprime un indicateur ayant le nom spécifié dans la fenêtre spécifiée du graphique.

```
bool  IndicatorDelete(
   int            sub_win       // numéro de la sous-fenêtre
   const string   name          // nom court de l'indicateur
   );
```

Paramètres

sub_win

[in] Numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre du graphique principal.

const name

[in] Le nom court de l'indicateur qui est spécifié dans la propriété INDICATOR_SHORTNAME avec la fonction IndicatorSetString(). Pour obtenir le nom court d'un indicateur, utiliser la fonction IndicatorName() .

Valeur de Retour

Retourne vrai si l'indicateur a été supprimé avec succès. Sinon retourne faux. Pour obtenir les détails de l'erreur, utilisez la fonction GetLastError().

Note

Si deux indicateurs avec des noms courts identiques existent dans la sous-fenêtre du graphique, le premier sera supprimé.

Si d'autres indicateurs sur ce graphique sont basés sur les valeurs de l'indicateur en cours de suppression, ces indicateurs seront supprimés.

Ne pas confondre le nom court de l'indicateur et le nom du fichier spécifié lors de la création d'un indicateur en utilisant les fonctions iCustom() et IndicatorCreate(). Si le nom court d'un indicateur n'est pas défini explicitement, alors le nom du fichier contenant le code source de l'indicateur sera défini pendant la compilation.

La suppression d'un indicateur d'un graphique ne signifie pas que les calculs seront supprimés de la mémoire du terminal. Pour relâcher le handle de l'indicateur, utiliser la fonction IndicatorRelease().

Le nom court de l'indicateur doit être formé correctement. Il sera affecté à la propriété INDICATOR_SHORTNAME avec la fonction IndicatorSetString(). Il est recommendé que le nom court contienne tous les paramètres d'entrée de l'indicateur, car l'indicateur à supprimer du graphique par la fonction IndicatorDelete() est identifié par son nom court.

Voir aussi

IndicatorAdd(), IndicatorsTotal(), IndicatorName(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2015, MetaQuotes Software Corp.

IndicatorsTotal

Retourne le nombre total d'indicateurs appliqués à la fenêtre du graphique spécifié.

```
int  IndicatorsTotal(
   int   sub_win     // numéro de la sous-fenêtre
   );
```

Paramètres

sub_win

[in] Numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre du graphique principal.

Valeur de Retour

Le nombre d'indicateurs dans la fenêtre spécifiée. Pour obtenir les détails de l'erreur, utilisez la fonction GetLastError().

Note

La fonction permet de rechercher parmi tous les indicateurs attachés au graphique. Le nombre total de fenêtres du graphique peut être obtenu depuis la propriété CHART_WINDOWS_TOTAL en utilisant la fonction GetInteger().

Voir aussi

IndicatorAdd(), IndicatorDelete(), IndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2015, MetaQuotes Software Corp.

IndicatorName

Retourne le nom court de l'indicateur par son index dans la liste des indicateurs de la fenêtre du graphique spécifié.

```
string  IndicatorName(
   int   sub_win     // numéro de la sous-fenêtre
   int   index       // index de l'indicateur dans la liste des indicateurs ajoutés à
   );
```

Paramètres

sub_win

[in] Numéro de la sous-fenêtre du graphique. 0 signifie la fenêtre du graphique principal.

index

[in] index de l'indicateur dans la liste des indicateurs. L'énumeration des indicateurs commence à zero, i.e. le premier indicateur de la liste à l'index 0. Pour obtenir le nombre d'indicateurs dans la liste, utiliser la fonction IndicatorsTotal().

Valeur de Retour

Le nom court de l'indicateur qui est affecté à la propriété INDICATOR_SHORTNAME avec la fonction IndicatorSetString(). Pour obtenir les détails de l'erreur, utiliser la fonction GetLastError().

Note

Ne pas confondre le nom court de l'indicateur et le nom du fichier spécifié lors de la création d'un indicateur en utilisant les fonctions iCustom() et IndicatorCreate(). Si le nom court d'un indicateur n'est pas défini explicitement, alors le nom du fichier contenant le code source de l'indicateur sera défini pendant la compilation.

Le nom court de l'indicateur doit être formé correctement. Il sera affecté à la propriété INDICATOR_SHORTNAME avec la fonction IndicatorSetString(). Il est recommendé que le nom court contienne tous les paramêtres d'entrée de l'indicateur, car l'indicateur à supprimer du graphique par la fonction IndicatorDelete() est identifié par son nom court.

Voir aussi

IndicatorAdd(), IndicatorDelete, IndicatorsTotal, iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2015, MetaQuotes Software Corp.

Navigate

Permet de se déplacer dans le graphique.

```
bool  Navigate(
   ENUM_CHART_POSITION  position,     // Position
   int                  shift=0       // Décalage
   )
```

Paramètres

position

[in] Valeur de l'énumération ENUM_CHART_POSITION.

shift=0

[in] Nombre de barres de décalage.

Valeur de retour

vrai si réalisé avec succès, faux si le graphique n'a pas été déplacé.

© 2000-2015, MetaQuotes Software Corp.

Symbol

Retourne le symbole du graphique.

```
string  Symbol() const
```

Valeur de retour

Symbole du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

Period

Retourne l'espace de temps du graphique.

```
ENUM_TIMEFRAMES  Period() const
```

Valeur de retour

L'espace de temps du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

Redraw

Redessine le graphique assigné à l'instance de classe.

```
void  Redraw()
```

© 2000-2015, MetaQuotes Software Corp.

GetInteger

Cette fonction retourne la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être de type integer (entier). Cette fonction possède 2 variantes.

1. Retourne immédiatement la valeur de la propriété.

```
long  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,          // identifiant de la propriété
   int                          sub_window=0      // numéro de la sous-fenêtre
   ) const
```

2. En cas de succès, affecte la valeur de la propriété dans la variable de type integer spécifiée et passée par référence comme dernier paramètre.

```
bool  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,        // identifiant de la propriété
   int                          sub_window,     // numéro de la sous-fenêtre
   long&                        value           // la valeur est récupérée ici
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumeration ENUM_CHART_PROPERTY_INTEGER).

sub_window

[in] Numéro de la sous-fenêtre du graphique.

value

[in] Variable de type integer qui reçoit la valeur de la propriété demandée.

Valeur de Retour

Valeur de la propriété du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne -1.

Pour la seconde variante de la fonction, retourne vrai si cette propriété est trouvée et que la valeur a été copiée dans la variable, sinon retourne faux. Pour en savoir plus sur les erreurs, utiliser la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

SetInteger

Définit la nouvelle valeur de la propriété de type integer.

```
bool  SetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,      // identifiant de la propriété
   long                         value         // nouvelle valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumeration ENUM_CHART_PROPERTY_INTEGER).

value

[in] Nouvelle valeur de la propriété.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété de type integer n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

GetDouble

Cette fonction retourne la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être de type double. Cette fonction possède 2 variantes.

1. Retourne immédiatement la valeur de la propriété.

```
double  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,          // identifiant de la propriété
   int                         sub_window=0     // numéro de la sous-fenêtre
   ) const
```

2. En cas de succès, affecte la valeur de la propriété dans la variable de type double spécifiée et passée par référence comme dernier paramètre.

```
bool  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,        // identifiant de la propriété
   int                         sub_window,     // numéro de la sous-fenêtre
   double&                     value           // la valeur est récupérée ici
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumération ENUM_CHART_PROPERTY_DOUBLE).

sub_window

[in] Numéro de la sous-fenêtre du graphique.

value

[in] Variable de type double qui reçoit la valeur de la propriété demandée.

Valeur de Retour

Valeur de la propriété du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne EMPTY_VALUE.

Pour la seconde variante de la fonction, retourne vrai si cette propriété est trouvée et que la valeur a été copiée dans la variable, sinon retourne faux. Pour en savoir plus sur les erreurs, utiliser la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

SetDouble

Définit la nouvelle valeur de la propriété de type double.

```
bool  SetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,      // identifiant de la propriété
   double                      value         // nouvelle valeur
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumération ENUM_CHART_PROPERTY_DOUBLE).

value

[in] Nouvelle valeur pour la propriété.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété de type double n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

GetString

Cette fonction retourne la valeur de la propriété correspondante de l'objet. La propriété de l'objet doit être de type string. Cette fonction possède 2 variantes.

1. Retourne immédiatement la valeur de la propriété.

```
string  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id      // identifiant de la propriété
   ) const
```

2. En cas de succès, affecte la valeur de la propriété dans la variable de type string spécifiée et passée par référence comme dernier paramètre.

```
bool  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,     // identifiant de la propriété
   string&                     value        // la valeur est récupérée ici
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumération ENUM_CHART_PROPERTY_STRING).

sub_window

[in] Numéro de la sous-fenêtre du graphique.

value

[in] Variable de type string qui reçoit la valeur de la propriété demandée.

Valeur de Retour

Valeur de la propriété du graphique assigné à l'instance de classe. Si aucun graphique n'est assigné, retourne "".

Pour la seconde variante de la fonction, retourne vrai si cette propriété est trouvée et que la valeur a été copiée dans la variable, sinon retourne faux. Pour en savoir plus sur les erreurs, utiliser la fonction GetLastError().

© 2000-2015, MetaQuotes Software Corp.

SetString

Définit la nouvelle valeur de la propriété de type string.

```
bool  SetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,      // identifiant de la propriété
   string                      value         // nouvelle valeur de la propriété
   )
```

Paramètres

prop_id

[in] Identifiant de la propriété (énumération ENUM_CHART_PROPERTY_STRING).

value

[in] Nouvelle valeur pour la propriété.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété de type string n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

SetSymbolPeriod

Change le symbole et l'espace de temps du graphique assignés à l'instance de classe.

```
bool  SetSymbolPeriod(
   const string      symbol_name,      // symbole
   ENUM_TIMEFRAMES   timeframe         // Période
   )
```

Paramètres

symbol_name

[in] Nom du nouveau symbole. NULL signifie le symbole du graphique courant (auquel l'expert est attaché).

timeframe

[in] Nouvelle espace de temps du graphique (énumération ENUM_TIMEFRAMES). 0 signifie l'espace de temps actuel.

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

ApplyTemplate

Applique le modèle spécifié au graphique.

```
bool  ApplyTemplate(
   const string  filename      // nom du fichier modèle
   )
```

Paramètres

filename

[in] Nom du fichier du modèle.

Valeur de retour

vrai si réalisé avec succès, faux si le modèle n'a pas été appliqué.

© 2000-2015, MetaQuotes Software Corp.

ScreenShot

Crée une copie d'écran du graphique spécifié et le sauvegarde dans un fichier .gif.

```
bool  ScreenShot(
   string           filename,                  // Nom de fichier
   int              width,                     // Largeur
   int              height,                    // Hauteur
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // Type d'alignement
   ) const
```

Paramètres

filename

[in] Nom du fichier destination de la copie d'écran.

width

[in] Largeur de la copie d'écran en pixels.

height

[in] Hauteur de la copie d'écran en pixels.

align_mode=ALIGN_RIGHT

[in] Mode d'alignement si la copie d'écran n'est pas assez large.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

WindowOnDropped

Retourne le numéro de la sous-fenêtre du graphique correspondant au point de dépose de l'objet (expert ou script).

```
int  WindowOnDropped() const
```

Valeur de retour

Numéro de la sous-fenêtre du graphique correspondant au point de dépose de l'objet. 0 signifie la fenêtre du graphique principal.

© 2000-2015, MetaQuotes Software Corp.

PriceOnDropped

Retourne le prix correspondant au point de dépose de l'objet (expert ou script).

```
double  PriceOnDropped() const
```

Valeur de retour

Le prix correspondant au point de dépose de l'objet.

© 2000-2015, MetaQuotes Software Corp.

TimeOnDropped

Retourne la date/heure correspondant au point de dépose de l'objet (expert ou script).

```
datetime  TimeOnDropped() const
```

Valeur de retour

Date/heure correspondant au point de dépose de l'objet.

© 2000-2015, MetaQuotes Software Corp.

XOnDropped

Retourne la coordonnée X correspondant au point de dépose de l'objet (expert ou script).

```
int  XOnDropped() const
```

Valeur de retour

Coordonnée X du point de dépose de l'objet.

© 2000-2015, MetaQuotes Software Corp.

YOnDropped

Retourne la coordonnée Y correspondant au point de dépose de l'objet (expert ou script).

```
int  YOnDropped() const
```

Valeur de retour

Coordonnée Y du point de dépose de l'objet.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les paramètres de l'objet dans un fichier.

```
virtual bool  Save(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire déjà ouvert avec la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les paramètres d'un objet depuis un fichier.

```
virtual bool  Load(
   int  file_handle      // Handle du fichier
   )
```

Paramètres

file_handle

[in] handle du fichier binaire déjà ouvert avec la fonction FileOpen(...).

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet graphique.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type d'objet (0x1111 pour CChart).

© 2000-2015, MetaQuotes Software Corp.

Opérations sur les Fichiers

Cette section contient les détails techniques d'utilisation des classes d'opérations sur les fichiers et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Utiliser les classes liées aux opérations sur les fichiers vous fera gagner du temps lors du développement d'applications qui utilisent les opérations d'entrée/sortie sur les fichiers.

La Bibliothèque Standard MQL5 (en terme de données) est située dans le répertoire de travail du terminal dans le répertoire Include\Files.

Classe	Description
CFile	Classe de base pour les opérations sur les fichiers
CFileBin	Classe pour les opérations sur les fichiers binaires
CFileTxt	Classe pour les opérations sur les fichiers textes

© 2000-2015, MetaQuotes Software Corp.

CFile

CFile est la classe de base pour les classes CFileBin et CFileTxt.

Description

La classe CFile fournit un accès simplifié pour toutes ses classes dérivées aux fonctions relatives aux fichiers et aux répertoires de l'API MQL5.

Déclaration

```
class CFile: public CObject
```

Titre

```
#include <Files\File.mqh>
```

Méthodes de Classe

Attributs	
Handle	Retourne le handle du fichier
Filename	Retourne le nom du fichier
Flags	Retourne les flags du fichier
SetUnicode	Définit/Réinitialise le flag FILE_UNICODE
SetCommon	Définit/Réinitialise le flag FILE_COMMON
Méthodes générales liées aux fichiers	
Open	Ouvre un fichier
Close	Ferme un fichier
Delete	Supprime un fichier
IsExist	Vérifie l'existence d'un fichier
Copy	Copie un fichier
Move	Renomme/déplace un fichier
Size	Retourne la taille du fichier
Tell	Retourne la position courante dans le fichier
Seek	Définit la position courante dans le fichier
Flush	Ecrit les données sur le disque
IsEnding	Vérifie la fin du fichier
IsLineEnding	Vérifie la fin de ligne du fichier
Méthodes générales liées aux répertoires	

© 2000-2015, MetaQuotes Software Corp.

FolderCreate	Crée un répertoire
FolderDelete	Supprime un répertoire
FolderClean	Vide un répertoire
Méthodes de recherche	
FileFindFirst	Commence une recherche de fichier
FileFindNext	Continue une recherche de fichier
FileFindClose	Ferme la handle de recherche

Classes dérivées :

- CFileBin
- CFileTxt

© 2000-2015, MetaQuotes Software Corp.

Handle

Retourne le handle du fichier ouvert.

```
int  Handle()
```

Valeur de retour

Handle du fichier ouvert assigné à l'instance de classe. Si aucun fichier n'est assigné, retourne -1.

© 2000-2015, MetaQuotes Software Corp.

FileName

Retourne le nom du fichier ouvert.

```
string  FileName()
```

Valeur de retour

Nom du fichier ouvert assigné à l'instance de la classe. Si aucun fichier n'est assigné, retourne "".

© 2000-2015, MetaQuotes Software Corp.

Flags

Retourne les flags du fichier ouvert.

```
int  Flags()
```

Valeur de retour

Flags du fichier ouvert assigné à l'instance de la clase.

© 2000-2015, MetaQuotes Software Corp.

SetUnicode

Définit/Réinitialise le flag FILE_UNICODE.

```
void  SetUnicode(
   bool  unicode      // Nouvelle valeur du flag
   )
```

Paramètres

unicode

[in] Nouvelle valeur du flag FILE_UNICODE.

Note

Le résultat des opérations sur les chaînes de caractères dépend du flag FILE_UNICODE. S'il est faux, les codes ANSI sont utilisés (symboles sur 1 octet). S'il est défini, les codes UNICODE sont utilisés (symboles sur 2 octets). Si le fichier a déjà été ouvert, le flag ne peut pas être changé.

© 2000-2015, MetaQuotes Software Corp.

SetCommon

Définit/Réinitialise le flag FILE_COMMON.

```
void  SetCommon(
   bool  common      // Nouvelle valeur du flag
   )
```

Paramètres

common

[in] Nouvelle valeur du flag FILE_COMMON.

Note

La flag FILE_UNICODE détermine le répertoire courant de travail. S'il est faux, le répertoire local du terminal est utilisé comme répertoire courant de travail. S'il est vrai, le répértoire général est utilisé comme répertoire courant de travail. Si le fichier a déjà été ouvert, le flag ne peut pas être changé.

© 2000-2015, MetaQuotes Software Corp.

Open

Ouvre le fichier spécifié et l'assigne à l'instance de class en cas de succès.

```
int  Open(
   const string  file_name,        // nom du fichier
   int           flags,            // Flags
   short         delimiter=9       // Séparateur
   )
```

Paramètres

file_name

[in] Nom du fichier à ouvrir.

flags

[in] Flags d'ouverture du fichier.

delimiter=9

[in] Séparateur du fichier CSV.

Valeur de retour

Handle sur le fichier ouvert.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2015, MetaQuotes Software Corp.

Close

Ferme le fichier assigné à l'instance de la classe.

```
void  Close()
```

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime le fichier assigné à l'instance du fichier.

```
void  Delete()
```

Delete

Supprime le fichier spécifié.

```
void  Delete(
   const string  file_name      // Nom du fichier
   )
```

Paramètres

file_name

[in] Nom du fichier à supprimer.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2015, MetaQuotes Software Corp.

IsExist

Vérifie l'existence d'un fichier

```
bool  IsExist(
   const string  file_name      // Nom du fichier
   )
```

Paramètres

file_name

[in] Nom du fichier à vérifier.

Valeur de retour

vrai si le fichier existe.

© 2000-2015, MetaQuotes Software Corp.

Copy

Copie un fichier.

```
bool  Copy(
   const string  src_name,      // Nom du fichier source
   int           src_flag,      // Flag
   const string  dst_name,      // Nom du fichier destination
   int           dst_flags      // Flags
   )
```

Paramètres

src_name

[in] Nom du fichier source à copier.

src_flag

[in] Flags du fichier à copier (seul FILE_COMMON est utilisé).

dst_name

[in] Nom du fichier de destination.

dst_flags

[in] Flags du fichier de destination (seuls FILE_REWRITE et FILE_COMMON sont utilisés).

Valeur de retour

vrai si réalisé avec succès, faux si la copie a échoué.

© 2000-2015, MetaQuotes Software Corp.

Move

Renomme/déplace un fichier.

```
bool  Move(
   const string  src_name,      // Nom du fichier source
   int           src_flag,      // Flag
   const string  dst_name,      // Nom du fichier destination
   int           dst_flags      // Flags
   )
```

Paramètres

src_name

[in] Nom du fichier à déplacer.

src_flag

[in] Flags du fichier à copier (seul FILE_COMMON est utilisé).

dst_name

[in] Nom du fichier de destination.

dst_flags

[in] Flags du fichier de destination (seuls FILE_REWRITE et FILE_COMMON sont utilisés).

Valeur de retour

vrai si réalisé avec succès, faux si le déplacement du fichier a échoué.

© 2000-2015, MetaQuotes Software Corp.

Size

Retourne la taille du fichier en octets.

```
ulong  Size()
```

Valeur de retour

Taille du fichier en octets. Si aucun fichier n'est assigné, retourne ULONG_MAX.

© 2000-2015, MetaQuotes Software Corp.

Tell

Retourne la position courante dans le fichier.

```
ulong  Tell()
```

Valeur de retour

La position courante dans le fichier. Si aucun fichier n'est assigné, retourne ULONG_MAX.

© 2000-2015, MetaQuotes Software Corp.

Seek

Définit la position courante dans le fichier.

```
void  Seek(
   long                offset,     // Décalage
   ENUM_FILE_POSITION  origin      // Origine
   )
```

Paramètres

offset

[in] Décalage dans le fichier en octets (peut être négatif).

origin

[in] Origine du décalage.

Valeur de retour

vrai si réalisé avec succès, faux si la position dans le fichier n'a pas été changée.

© 2000-2015, MetaQuotes Software Corp.

Flush

Ecrit (vide le tampon) toutes les données du buffer du fichier en entrée/sortie sur le disque.

```
void  Flush()
```

© 2000-2015, MetaQuotes Software Corp.

IsEnding

Vérifie la fin du fichier. Utilisé pendant les opérations de lecture du fichier.

```
bool  IsEnding()
```

Valeur de retour

vrai si la fin du fichier est atteinte après la dernière lecture ou opération de recherche.

© 2000-2015, MetaQuotes Software Corp.

IsLineEnding

Vérifie la fin de ligne du fichier. Utilisé pendant les opérations de lecture du fichier.

```
bool  IsLineEnding()
```

Valeur de retour

vrai si la fin de ligne du fichier est atteinte après la dernière opération de lecture d'un fichier txt ou csv (caractères CR-LF).

© 2000-2015, MetaQuotes Software Corp.

FolderCreate

Crée un nouveau répertoire.

```
bool  FolderCreate(
   const string  folder_name      // Nom du répertoire
   )
```

Paramètres

folder_name

[in] Nom du répertoire à créer. Contient le chemin relatif du répertoire définit avec le flag FILE_COMMON.

Valeur de retour

vrai si réalisé avec succès, faux si le répertoire n'a pas été créé.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2015, MetaQuotes Software Corp.

FolderDelete

Supprime le répertoire spécifié.

```
bool  FolderDelete(
   const string  folder_name      // Nom du répertoire
   )
```

Paramètres

folder_name

[in] Nom du répertoire à supprimer. Contient le chemin relatif du répertoire définit avec le flag FILE_COMMON.

Valeur de retour

vrai si réalisé avec succès, faux si le répertoire n'a pas été supprimé.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2015, MetaQuotes Software Corp.

FolderClean

Vide le répertoire sépcifié.

```
bool  FolderClean(
   const string  folder_name      // Nom du répertoire
   )
```

Paramètres

folder_name

[in] Nom du répertoire à supprimer. Contient le chemin relatif du répertoire définit avec le flag FILE_COMMON.

Valeur de retour

vrai si réalisé avec succès, faux si le répertoire n'a pas été vidé.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2015, MetaQuotes Software Corp.

FileFindFirst

Commence la recherche de fichier en utilisant le filtre spécifié.

```
int  FileFindFirst(
   const string  filter,        // Filtre de recherche
   string&       file_name      // Référence sur une chaîne de caractères
   )
```

Paramètres

filter

[in] Filtre de recherche.

file_name

[out] Référence sur une chaîne de caractères pour le premier fichier trouvé.

Valeur de retour

En cas de succès, retourne le handle qui peut être utilisé pour continuer la recherche de fichier en utilisant FileFindNext, ou INVALID_HANDLE si aucun fichier ne correspond au filtre spécifié.

Note

Le répertoire de travail dépend du flag FILE_COMMON, défini par la méthode SetCommon().

© 2000-2015, MetaQuotes Software Corp.

FileFindNext

Continue la recherche, commencée avec la fonction FileFindFirst().

```
bool  FileFindNext(
   int      search_handle,     // Handle de recherche
   string&  file_name          // Référence sur la chaîne de caractères pour le procha
   )
```

Paramètres

search_handle

[in] Handle de recherche, retourné par la méthode FileFindFirst().

file_name

[out] Référence sur la chaîne de caractères si un fichier est trouvé.

Valeur de retour

vrai en cas de succès, faux si aucun fichier correspondant au filtre spécifié n'est trouvé.

© 2000-2015, MetaQuotes Software Corp.

FileFindClose

Ferme la handle du fichier.

```
void  FileFindClose(
   int  search_handle      // Handle de recherche
   )
```

Paramètres

search_handle

[in] Handle de recherche, retourné par la méthode FileFindFirst().

© 2000-2015, MetaQuotes Software Corp.

CFileBin

CFileBin est une classe permettant un accès simplifié aux fichiers binaires.

Description

La classe CFileBin fournit un accès aux fichiers binaires.

Déclaration

```
class CFileBin: public CFile
```

Titre

```
#include <Files\FileBin.mqh>
```

Méthodes de Classe

Méthodes d'ouverture	
Open	Ouvre un fichier binaire
Méthodes d'écriture	
WriteChar	Ecrit une variable de type char ou uchar
WriteShort	Ecrit une variable de type short ou ushort
WriteInteger	Ecrit une variable de type int ou uint
WriteLong	Ecrit une variable de type long ou ulong
WriteFloat	Ecrit une variable de type float
WriteDouble	Ecrit une variable de type double
WriteString	Ecrit une variable de type string
WriteCharArray	Ecrit un tableau de variables de type char ou uchar
WriteShortArray	Ecrit un tableau de variables de type short ou ushort
WriteIntegerArray	Ecrit un tableau de variables de type int ou uint
WriteLongArray	Ecrit un tableau de variables de type long ou ulong
WriteFloatArray	Ecrit un tableau de variables de type float
WriteDoubleArray	Ecrit un tableau de variables de type double
WriteObject	Ecrit les données d'une instance dérivant de la classe CObject
Méthodes de lecture	

© 2000-2015, MetaQuotes Software Corp.

ReadChar	Lit une variable de type char ou uchar
ReadShort	Lit une variable de type short ou ushort
ReadInteger	Lit une variable de type int ou uint
ReadLong	Lit une variable de type long ou ulong
ReadFloat	Lit une variable de type float
ReadDouble	Lit une variable de type double
ReadString	Lit une variable de type string
ReadCharArray	Lit un tableau de variables de type char ou uchar
ReadShortArray	Lit un tableau de variables de type short ou ushort
ReadIntegerArray	Lit un tableau de variables de type int ou uint
ReadLongArray	Lit un tableau de variables de type long ou ulong
ReadFloatArray	Lit un tableau de variables de type float
ReadDoubleArray	Lit un tableau de variables de type double
ReadObject	Lit les données d'une instance de classe dérivant de la classe CObject

© 2000-2015, MetaQuotes Software Corp.

Open

Ouvre le fichier binaire spécifié et l'assigne à l'instance de classe en cas de succès.

```
int  Open(
   const string  file_name,      // Nom du fichier
   int           flags           // Flags
   )
```

Paramètres

file_name

[in] Nom du fichier à ouvrir.

flags

[in] Flags d'ouverture du fichier (flag FILE_BIN).

Valeur de retour

Handle sur le fichier ouvert.

© 2000-2015, MetaQuotes Software Corp.

WriteChar

Ecrit une variable de type char ou uchar dans le fichier.

```
uint  WriteChar(
   char  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteShort

Ecrit une variable de type short ou ushort dans le fichier.

```
uint  WriteShort(
   short  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteInteger

Ecrit une variable de type int ou uint dans le fichier.

```
uint  WriteInteger(
   int  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteLong

Ecrit une variable de type long ou ulong dans le fichier.

```
uint  WriteLong(
   long  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteFloat

Ecrit une variable de type float dans le fichier.

```
uint  WriteFloat(
   float  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteDouble

Ecrit une variable de type double dans le fichier.

```
uint  WriteDouble(
   double  value      // Valeur
   )
```

Paramètres

value

[in] Variable à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteString

Ecrit une variable de type string dans le fichier.

```
uint  WriteString(
   const string  value      // Valeur
   )
```

Paramètres

value

[in] Chaîne de caractères à écrire.

Valeur de retour

Nombre d'octets écrits.

WriteString

Ecrit une variable de type string dans le fichier.

```
uint  WriteString(
   const string  value,     // Valeur
   int           size       // Taille
   )
```

Paramètres

value

[in] Chaîne de caractères à écrire.

size

[in] Nombre d'octets à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteCharArray

Ecrit un tableau de variables de type char ou uchar dans le fichier.

```
uint  WriteCharArray(
   char&  array[],            // Référence sur le tableau
   int    start_item=0,       // Elément de départ
   int    items_count=-1      // Number of elements
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 - pour le tableau entier).

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteShortArray

Ecrit un tableau de variables de type short ou ushort dans le fichier.

```
uint  WriteShortArray(
   short&  array[],            // Tableau à écrire
   int     start_item=0,       // Elément de départ
   int     items_count=-1      // Nombre d'éléments à écrite (-1 pour le tableau compl
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 - pour le tableau entier).

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteIntegerArray

Ecrit un tableau de variables de type int ou uint dans le fichier.

```
uint  WriteIntegerArray(
   int&  array[],            // Tableau à écrire
   int   start_item=0,       // Elément de départ
   int   items_count=-1      // Nombre d'éléments à écrite (-1 pour le tableau complet
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 - pour le tableau entier).

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteLongArray

Ecrit un tableau de variables de type long ou <t3>ulong</t3><t4> dans le fichier.</t4>

```
uint  WriteLongArray(
   long&  array[],            // Tableau à écrire
   int    start_item=0,       // Elément de départ
   int    items_count=-1      // Nombre d'éléments à écrite (-1 pour le tableau comple
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 pour le tableau complet). (-1 - for whole array).

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteFloatArray

Ecrit un tableau de variables de type float dans le fichier.

```
uint  WriteFloatArray(
   float&  array[],            // Tableau à écrire
   int     start_item=0,       // Elément de départ
   int     items_count=-1      // Nombre d'éléments à écrite (-1 pour le tableau compl
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 pour le tableau complet). (-1 - for whole array).

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteDoubleArray

Ecrit un tableau de variables de type double dans le fichier.

```
uint  WriteDoubleArray(
   double&  array[],            // Tableau à écrire
   int      start_item=0,       // Elément de départ
   int      items_count=-1      // Nombre d'éléments à écrite (-1 pour le tableau comp
   )
```

Paramètres

array[]

[in] Tableau à écrire.

start_item=0

[in] Elément de départ to write from.

items_count=-1

[in] Nombre d'éléments à écrite (-1 pour le tableau complet). (-1 - for whole array).

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

WriteObject

Ecrit les données d'une instance de classe dérivant de la classe CObject dans le fichier.

```
bool  WriteObject(
   CObject*  object      // Référence sur l'objet
   )
```

Paramètres

object

[in] Référence sur l'instance à écrire de la classe dérivant de la classe CObject.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été écrite.

© 2000-2015, MetaQuotes Software Corp.

ReadChar

Lit une variable de type char ou uchar depuis le fichier.

```
bool  ReadChar(
   char&  value      // Variable destination
   )
```

Paramètres

value

[in] Variable destination de type char.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadShort

Lit une variable de type short ou ushort depuis le fichier.

```
bool  ReadShort(
   short&  value
   )
```

Paramètres

value

[in] Variable destination de type short ou ushort.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadInteger

Lit une variable de type int ou uint depuis le fichier.

```
bool  ReadInteger(
   int&  value      // Variable destination
   )
```

Paramètres

value

[in] Variable destination de type int ou uint.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadLong

Lit une variable de type long ou ulong depuis le fichier.

```
bool  ReadLong(
   long&  value
   )
```

Paramètres

value

[in] Variable destination de type long ou ulong.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadFloat

Lit une variable de type float depuis le fichier.

```
bool  ReadFloat(
   float&  value      // Variable destination
   )
```

Paramètres

value

[in] Variable destination de type float.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadDouble

Lit une variable de type double depuis le fichier.

```
bool  ReadDouble(
   double&  value
   )
```

Paramètres

value

[in] Variable destination de type double.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadString

Lit une variable de type string depuis le fichier.

```
bool  ReadString(
   string&  value      // Chaîne desintation
   )
```

Paramètres

value

[in] Variable destination de type string.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

ReadString

Lit une variable de type string depuis le fichier.

```
bool  ReadString(
   string&  value
   )
```

Paramètres

value

[in] Variable destination de type string.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadCharArray

Lit un tableau de variables de type char ou uchar depuis le fichier.

```
bool  ReadCharArray(
   char&  array[],            // Tableau destination
   int    start_item=0,       // Elément de départ
   int    items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type char ou uchar.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadShortArray

Lit un tableau de variables de type short ou ushort depuis le fichier.

```
bool  ReadShortArray(
   short&  array[],            // Tableau destination
   int     start_item=0,       // Elément de départ
   int     items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type short ou ushort.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadIntegerArray

Lit un tableau de variables de type int ou uint depuis le fichier.

```
bool  ReadIntegerArray(
   int&  array[],            // Tableau destination
   int   start_item=0,       // Elément de départ
   int   items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type int ou uint.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadLongArray

Lit un tableau de variables de type long ou ulong depuis le fichier.

```
bool  ReadLongArray(
   long&  array[],            // Tableau destination
   int    start_item=0,       // Elément de départ
   int    items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type long ou ulong.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadFloatArray

Lit un tableau de variables de type float depuis le fichier.

```
bool  ReadFloatArray(
   float&  array[],            // Tableau destination
   int     start_item=0,       // Elément de départ
   int     items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type float.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadDoubleArray

Lit un tableau de variables de type double depuis le fichier.

```
bool  ReadDoubleArray(
   double&  array[],            // Tableau destination
   int      start_item=0,       // Elément de départ
   int      items_count=-1      // Nombre d'éléments à lire
   )
```

Paramètres

array[]

[in] Référence sur le tableau destination de type double.

start_item=0

[in] Elément de départ.

items_count=-1

[in] Nombre d'éléments à lire (-1 - lit jusqu'à la fin du fichier).

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

ReadObject

Lit les données d'une instance de classe dérivant de la classe CObject depuis le fichier.

```
bool  ReadObject(
   CObject*  object      // Référence sur l'objet
   )
```

Paramètres

object

[in] Référence sur l'instance à lire de la classe dérivant de la classe CObject.

Valeur de retour

vrai si réalisé avec succès, faux si la donnée n'a pas été lue.

© 2000-2015, MetaQuotes Software Corp.

CFileTxt

CFileTxt est une classe permettant un accès simplifié aux fichiers textes.

Description

La classe CFileTxt fournit un accès aux fichiers textes.

Déclaration

```
class CFileTxt: public CFile
```

Titre

```
#include <Files\FileTxt.mqh>
```

Méthodes de Classe

Méthodes d'ouverture	
Open	Ouvre un fichier texte
Méthodes d'écriture	
WriteString	Ecrit une variable de type string dans le fichier
Méthodes de lecture	
ReadString	Lit une variable de type string depuis le fichier.

© 2000-2015, MetaQuotes Software Corp.

Open

Ouvre le fichier texte spécifié et l'assigne à l'instance de classe en cas de succès.

```
int  Open(
   const string  file_name,     // nom du fichier
   int           flags          // flags
   )
```

Paramètres

file_name

[in] Nom du fichier à ouvrir.

flags

[in] Flags d'ouverture du fichier (flag FILE_TXT).

Valeur de retour

Handle du fichier ouvert.

© 2000-2015, MetaQuotes Software Corp.

WriteString

Ecrit une variable de type string dans le fichier.

```
uint  WriteString(
   const string  value      // Chaîne de caractères à écrire
   )
```

Paramètres

value

[in] Chaîne de caractères à écrire.

Valeur de retour

Nombre d'octets écrits.

© 2000-2015, MetaQuotes Software Corp.

ReadString

Lit une variable de type string depuis le fichier.

```
string  ReadString()
```

Valeur de retour

Chaîne de caractères qui a été lue.

© 2000-2015, MetaQuotes Software Corp.

Opérations sur les chaînes de caractères

Cette section contient les détails techniques d'utilisation des classes d'opérations sur les chaînes de caractères et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Utiliser les classes liées aux opérations sur les chaînes de caractères vous fera gagner du temps lors du développement d'applications qui utilisent les opérations de traitement de chaînes de caractères.

La Bibliothèque Standard MQL5 est située dans le répertoire de travail du terminal dans le répertoire Include\Strings.

Classe	Description
CString	Classe pour les opérations sur les chaines de caractères

© 2000-2015, MetaQuotes Software Corp.

CString

CString est une classe permettant un accès simplifié aux variables de type string.

Description

La classe CFile fournit un accès simplifié à tous ses descendants aux fonctions de chaînes de caractères de l'API MQL5.

Déclaration

```
class CString: public CObject
```

Titre

```
#include <Strings\String.mqh>
```

Méthodes de Classe

Méthodes d'accès aux données	
Str	Retourne la chaîne de caractères
Len	Retourne la longueur de la chaîne de caractères
Copy	Copie une chaîne de caractères
Méthodes de remplissage	
Fill	Remplit une chaîne de caractères avec un caractère spécifié
Assign	Assigne une chaîne de caractères
Append	Ajoute une chaîne de caractères
Insert	Insère une chaîne de caractères
Méthodes de comparaison	
Compare	Compare une chaîne de caractères
CompareNoCase	Compare une chaîne de caractères sans tenir compte de la casse
Méthodes de sous-chaînes de caractères	
Left	Retourne le nombre spécifié de caractères depuis la gauche de la chaîne de caractères
Right	Retourne le nombre spécifié de caractères depuis la droite de la chaîne de caractères
Mid	Retourne le nombre spécifié de caractères de la chaîne de caractères
Méthodes d'effacement/suppression	

© 2000-2015, MetaQuotes Software Corp.

Trim	Supprime tous les ensembles de caractères spécifiés devant et derrière la chaîne de caractères
TrimLeft	Supprime tous les ensembles de caractères spécifiés devant la chaîne de caractères
TrimRight	Supprime tous les ensembles de caractères spécifiés derrière la chaîne de caractères
Clear	Vide la chaîne de caractères
Méthodes de conversion	
ToUpper	Transforme une chaîne de caractères en majuscules.
ToLower	Transforme une chaîne de caractères en minuscules.
Reverse	Retourne une chaîne de caractères
Méthodes de recherche	
Find	Recherche la première occurence d'une chaîne de caractères
FindRev	Recherche la dernière occurence d'une chaîne de caractères
Remove	Supprime une sous-chaîne de caractères de la chaîne de caractères
Replace	Remplace une sous-chaîne de caractères

© 2000-2015, MetaQuotes Software Corp.

Str

Retourne la chaîne de caractères.

```
string  Str() const;
```

Valeur de retour

Copie de la chaîne de caractères.

© 2000-2015, MetaQuotes Software Corp.

Len

Retourne la longueur de la chaîne de caractères.

```
uint  Len() const;
```

Valeur de retour

Longueur de la chaîne de caractères.

© 2000-2015, MetaQuotes Software Corp.

Copy

Copie une chaîne de caractères par référence.

```
void  Copy(
   string&  copy      // Référence
   ) const;
```

Paramètres

copy

[in] Référence sur une chaîne de caractères à copier.

Copy

Copie une chaîne de caractères dans l'instance de classe CString.

```
void  Copy(
   CString*  copy      // Descripteur de l'objet
   ) const;
```

Paramètres

copy

[in] Descripteur de l'objet de classe CString.

© 2000-2015, MetaQuotes Software Corp.

Fill

Remplit une chaîne de caractères avec le caractère spécifié.

```
bool  Fill(
   short  character      // Caractère
   )
```

Paramètres

character

[in] Caractère de remplissage.

Valeur de retour

vrai si réalisé avec succès, faux si la chaîne de caractères n'a pas pu être remplie.

© 2000-2015, MetaQuotes Software Corp.

Assign

Assigne une chaîne de caractères.

```
void  Assign(
   const string  str      // Chaîne de caractères à assigner
   )
```

Paramètres

str

[in] Chaîne de caractères à assigner.

Assign

AjotueAssigne une chaîne de caractères à l'instance de classe CString.

```
void  Assign(
   CString*  str      // Descripteur de l'objet
   )
```

Paramètres

str

[in] Descripteur de l'objet de classe CString à assigner.

© 2000-2015, MetaQuotes Software Corp.

Append

Ajoute une chaîne de caractères à la fin.

```
void  Append(
   const string  str      // Chaîne de caractères à ajouter
   )
```

Paramètres

str

[in] Chaîne de caractères à ajouter.

Append

Ajoute une chaîne de caractères à l'instance de classe CString.

```
void  Append(
   CString*  string      // Descripteur de l'objet
   )
```

Paramètres

string

[in] Descripteur de l'objet de classe CString à ajouter.

© 2000-2015, MetaQuotes Software Corp.

Insert

Insère une chaîne de caractères à la position spécifiée.

```
uint  Insert(
   uint          pos,     // Position
   const string  str      // Chaîne de caractères à insérer
   )
```

Paramètres

pos

[in] Position d'insertion.

str

[in] Chaîne de caractères à insérer.

Valeur de retour

Taille de la chaîne de caractères résultante.

Insert

Insère une chaîne de caractères à la position spécifiée dans l'instance de classe CString.

```
uint  Insert(
   uint      pos,     // Position
   CString*  str      // Descripteur de l'objet
   )
```

Paramètres

pos

[in] Position d'insertion.

str

[in] Descripteur de l'objet de classe CString à insérer.

Valeur de retour

Taille de la chaîne de caractères résultante.

© 2000-2015, MetaQuotes Software Corp.

Compare

Compare une chaîne de caractères.

```
int  Compare(
   const string  str      // Chaîne de caractères à comparer
   ) const;
```

Paramètres

str

[in] Chaîne de caractères à comparer.

Valeur de retour

Retourne 0 si les chaînes de caractères sont identiques, -1 si la chaîne de caractères de la classe est inférieure à la chaîne de comparaison, 1 si la chaîne de caractères de la classe est supérieure à la chaîne de comparaison.

Compare

Compare une chaîne de caractères avec une chaîne de caractères de l'instance de class CString.

```
int  Compare(
   CString*  str      // Descripteur de l'objet
   ) const;
```

Paramètres

str

[in] Descripteur de l'objet de classe CString à comparer.

Valeur de retour

Retourne 0 si les chaînes de caractères sont identiques, -1 si la chaîne de caractères de la classe est inférieure à la chaîne de comparaison, 1 si la chaîne de caractères de la classe est supérieure à la chaîne de comparaison.

© 2000-2015, MetaQuotes Software Corp.

CompareNoCase

Compare une chaîne de caractères sans tenir compte de la casse.

```
int  CompareNoCase(
   const string  str      // Chaîne de caractères à comparer
   ) const;
```

Paramètres

str

[in] Chaîne de caractères à comparer.

Valeur de retour

Retourne 0 si les chaînes de caractères sont identiques, -1 si la chaîne de caractères de la classe est inférieure à la chaîne de comparaison, 1 si la chaîne de caractères de la classe est supérieure à la chaîne de comparaison.

CompareNoCase

Compare une chaîne de caractères (sans tenir compte de la casse) avec une chaîne de caractères de l'instance de class CString.

```
int  CompareNoCase(
   CString*  str      // Descripteur de l'objet
   ) const;
```

Paramètres

str

[in] Descripteur de l'objet de classe CString à comparer.

Valeur de retour

Retourne 0 si les chaînes de caractères sont identiques, -1 si la chaîne de caractères de la classe est inférieure à la chaîne de comparaison, 1 si la chaîne de caractères de la classe est supérieure à la chaîne de comparaison.

© 2000-2015, MetaQuotes Software Corp.

Left

Retourne le nombre spécifié de caractères depuis la gauche de la chaîne de caractères.

```
string  Left(
   uint  count      // Nombre de caractères
   )
```

Paramètres

count

[in] Nombre de caractères.

Valeur de retour

Chaîne de caractères résultante.

© 2000-2015, MetaQuotes Software Corp.

Right

Retourne le nombre spécifié de caractères depuis la droite de la chaîne de caractères.

```
string  Right(
   uint  count      // Nombre de caractères
   )
```

Paramètres

count

[in] Nombre de caractères.

Valeur de retour

Chaîne de caractères résultante.

© 2000-2015, MetaQuotes Software Corp.

Mid

Retourne le nombre spécifié de caractères de la chaîne de caractères.

```
string  Mid(
   uint  pos,        // Position
   uint  count       // Nombre de caractères
   )
```

Paramètres

pos

[in] Position de la chaîne de caractères.

count

[in] Nombre de caractères.

Valeur de retour

Chaîne de caractères résultante.

© 2000-2015, MetaQuotes Software Corp.

Trim

Supprime toutes les occurences d'un ensemble de caractères spécifiés (et ' ','\t','\r','\n') avant et après la chaîne de caractères.

```
int  Trim(
   const string  targets      // Ensemble de caractères à supprimer
   )
```

Paramètres

targets

[in] Ensemble de caractères à supprimer.

Valeur de retour

Nombre de caractères supprimés.

Exemple :

```
//--- exemple d'utilisation de CString::Trim
#include <Strings\String.mqh>
//---
void OnStart()
  {
   CString str;
   //---
   str.Assign("   \t\tABCD\r\n");
   printf("Source string '%s'",str.Str());
   //---
   str.Trim("DA-DA-DA");
   printf("Result string '%s'",str.Str());
  }
```

© 2000-2015, MetaQuotes Software Corp.

TrimLeft

Supprime toutes les occurences d'un ensemble de caractères spécifiés (et ' ','\t','\r','\n') au début de la chaîne de caractères.

```
int  TrimLeft(
   const string  targets      // Ensemble de caractères à supprimer
   )
```

Paramètres

targets

[in] Ensemble de caractères à supprimer.

Valeur de retour

Nombre de caractères supprimés.

© 2000-2015, MetaQuotes Software Corp.

TrimRight

Supprime toutes les occurences d'un ensemble de caractères spécifiés (et ' ','\t','\r','\n') à la fin de la chaîne de caractères.

```
int  TrimRight(
   const string  targets      // Ensemble de caractères à supprimer
   )
```

Paramètres

targets

[in] Ensemble de caractères à supprimer.

Valeur de retour

Nombre de caractères supprimés.

© 2000-2015, MetaQuotes Software Corp.

Clear

Vide la chaîne de caractères.

```
bool  Clear()
```

Valeur de retour

vrai si réalisé avec succès, faux si la chaîne de caractères ne peut pas être vidée.

© 2000-2015, MetaQuotes Software Corp.

ToUpper

Transforme une chaîne de caractères en majuscules.

```
bool  ToUpper()
```

Valeur de retour

vrai si réalisé avec succès, faux si la chaîne de caractères n'a pas pu être convertie.

© 2000-2015, MetaQuotes Software Corp.

ToLower

Transforme une chaîne de caractères en minuscules.

```
bool  ToLower()
```

Valeur de retour

vrai si réalisé avec succès, faux si la chaîne de caractères n'a pas pu être convertie.

© 2000-2015, MetaQuotes Software Corp.

Reverse

Retourne une chaîne de caractères.

```
void  Reverse()
```

© 2000-2015, MetaQuotes Software Corp.

Find

Recherche la première occurence d'une chaîne de caractères.

```
int  Find(
   uint          start,          // Position
   const string  substring       // Chaîne de caractères à rechercher
   ) const;
```

Paramètres

start

[in] L'index du caractère de la chaîne correspondant à l'index de départ de la recherche, ou 0 pour commencer au début.

substring

[in] Chaîne de caractères à rechercher.

Valeur de retour

L'index du premier caractère correspondant à la chaîne recherchée, ou -1 si la chaîne de caractères n'a pas été trouvée.

© 2000-2015, MetaQuotes Software Corp.

FindRev

Recherche la dernière occurence d'une chaîne de caractères.

```
int  FindRev(
   const string  substring      // Chaîne de caractères à rechercher
   ) const;
```

Paramètres

substring

[in] Chaîne de caractères à rechercher.

Valeur de retour

L'index du dernier caractère correspondant à la chaîne recherchée, ou -1 si la chaîne de caractères n'a pas été trouvée.

© 2000-2015, MetaQuotes Software Corp.

Remove

Supprime une sous-chaîne de caractères de la chaîne de caractères.

```
uint  Remove(
   const string  substring      // Sous-chaîne de caractères à supprimer
   )
```

Paramètres

substring

[in] Chaîne de caractères à rechercher.

Valeur de retour

Nombre de sous-chaînes supprimées.

© 2000-2015, MetaQuotes Software Corp.

Replace

Remplace une sous-chaîne de la chaîne de caractères.

```
uint  Replace(
   const string  substring,     // Sous-chaîne à remplacer
   const string  newstring      // Nouvelle sous-chaîne
   )
```

Paramètres

substring

[in] Chaîne de caractères à rechercher.

newstring

[in] Sous-chaîne de remplacement.

Valeur de retour

Nombre de sous-chaînes remplacées.

© 2000-2015, MetaQuotes Software Corp.

Indicateurs Techniques et Séries de Données

Cette section contient les détails techniques d'utilisation des indicateurs techniques et des séries de données, ainsi qu'une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation des classes des indicateurs techniques et des séries de données classes vous fera gagner du temps lors du dévelopement d'applications (scripts, Experts Advisor).

La Bibliothèque Standard MQL5 est située dans le répertoire de travail du terminal dans le répertoire Include\Indicators.

Classe/groupe	Description
Classes de base	Groupe de base et classes annexes
Classes des séries de données	Groupe de classes des séries de données
Indicateurs de Tendance	Groupe de classes d'indicateurs de Tendance
Oscillateurs	Groupe de classes d'Oscillateurs
Indicateurs de Volumes	Groupe de classes d'indicateurs de Volume
Indicateurs de Bill Williams	Classes d'indicateurs de Bill Williams
Indicateurs personnalisés	Classe d'indicateurs personnalisés

© 2000-2015, MetaQuotes Software Corp.

Classes de Base et Classes Annexes des Indicateurs Techniques et des Séries de Données

Cette section contient les détails techniques des classes de base et des classes annexes des indicateurs techniques et des séries de données, ainsi qu'une description des composants correspondant de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

Classe/groupe	Description
CSpreadBuffer	Classe du buffer de l'historique du spread
CTimeBuffer	Classe du buffer de l'historique des prix d'ouverture
CTickVolumeBuffer	Classe du buffer de l'historique des volumes des ticks
CRealVolumeBuffer	Classe du buffer de l'historique des volumes réels
CDoubleBuffer	Classe de base du buffer de données de type double
COpenBuffer	Classe du buffer des prix d'ouverture
CHighBuffer	Classe du buffer des prix les plus hauts
CLowBuffer	Classe du buffer des prix les plus bas
CCloseBuffer	Classe du buffer des prix de clôture
CIndicatorBuffer	Classe du buffer de l'indicateur technique
CSeries	Classe de base pour l'accès aux données des séries de données
CPriceSeries	Classe de base pour l'accès aux prix
CIndicator	Classe de base d'un indicateur technique
CIndicators	Collection d'indicateurs techniques et de séries de données

© 2000-2015, MetaQuotes Software Corp.

CSpreadBuffer

CSpreadBuffer est une classe permettant un accès simplifié aux spreads des barres de l'historique.

Description

La classe CSpreadBuffer fournit l'accès aux spreads des barres de l'historique.

Déclaration

```
class CSpreadBuffer: public CArrayInt
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Attributs	
Size	Définit la taille du buffer
Paramètres	
SetSymbolPeriod	Définit le symbole et la période
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer par son index.
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

© 2000-2015, MetaQuotes Software Corp.

Size

Définit la taille du buffer.

```
void  Size(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2015, MetaQuotes Software Corp.

SetSymbolPeriod

Définit le symbole et la période.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbole
   const ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Nouveau symbole.

period

[in] Nouvelle période (énumération ENUM_TIMEFRAMES).

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du buffer par son index.

```
int  At(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

CTimeBuffer

CTimeBuffer est une classe permettant un accès simplifié aux prix d'ouverture des barres de l'historique.

Description

La classe CTimeBuffer fournit l'accès aux prix d'ouverture des barres de l'historique.

Déclaration

```
class CTimeBuffer: public CArrayLong
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Attributs	
Size	Définit la taille du buffer
Paramètres	
SetSymbolPeriod	Définit le symbole et la période
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer par son index.
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

© 2000-2015, MetaQuotes Software Corp.

Size

Définit la taille du buffer.

```
void  Size(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2015, MetaQuotes Software Corp.

SetSymbolPeriod

Définit le symbole et la période.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbole
   const ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Nouveau symbole.

period

[in] Nouvelle période (énumération ENUM_TIMEFRAMES).

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du buffer par son index.

```
long  At(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

CTickVolumeBuffer

CTickVolumeBuffer est une classe permettant un accès simplifié aux volumes des ticks des barres de l'historique.

Description

La classe CTickVolumeBuffer fournit l'accès aux volumes des ticks des barres de l'historique.

Déclaration

```
class CTickVolumeBuffer: public CArrayLong
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Attributs	
Size	Définit la taille du buffer
Paramètres	
SetSymbolPeriod	Définit le symbole et la période
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer par son index.
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

© 2000-2015, MetaQuotes Software Corp.

Size

Définit la taille du buffer.

```
void  Size(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2015, MetaQuotes Software Corp.

SetSymbolPeriod

Définit le symbole et la période.

```
void  SetSymbolPeriod(
   const string           symbol,      // symbole
   const ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Nouveau symbole.

period

[in] Nouvelle période (énumération ENUM_TIMEFRAMES).

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du buffer par son index.

```
long  At(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

CRealVolumeBuffer

CCloseBuffer est une classe permettant un accès simplifié aux volumes rééls des barres de l'historique.

Description

La classe CTickVolumeBuffer fournit l'accès aux volumes rééls des barres de l'historique.

Déclaration

```
class CRealVolumeBuffer: public CArrayLong
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Attributs	
Size	Définit la taille du buffer
Paramètres	
SetSymbolPeriod	Définit le symbole et la période
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer par son index.
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

© 2000-2015, MetaQuotes Software Corp.

Size

Définit la taille du buffer.

```
void  Size(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2015, MetaQuotes Software Corp.

SetSymbolPeriod

Définit le symbole et la période.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbole
   const ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Nouveau symbole.

period

[in] Nouvelle période (énumération ENUM_TIMEFRAMES).

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du buffer par son index.

```
long  At(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

CDoubleBuffer

CDoubleBuffer est une classe de base permettant un accès simplifié aux buffers de données de type double.

Description

La classe CDoubleBuffer fournit l'accès aux données de type double du buffer.

Déclaration

```
class CDoubleBuffer: public CArrayDouble
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Attributs	
Size	Définit la taille du buffer
Paramètres	
SetSymbolPeriod	Définit le symbole et la période
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

© 2000-2015, MetaQuotes Software Corp.

Size

Définit la taille du buffer.

```
void  Size(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2015, MetaQuotes Software Corp.

SetSymbolPeriod

Définit le symbole et la période.

```
void  SetSymbolPeriod(
   const string           symbol,      // symbole
   const ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Nouveau symbole.

period

[in] Nouvelle période (énumération ENUM_TIMEFRAMES).

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du buffer par son index.

```
double  At(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

COpenBuffer

COpenBuffer est une classe permettant un accès simplifié aux prix d'ouverture des barres dans l'historique.

Description

COpenBuffer est une classe permettant un accès simplifié aux prix d'ouverture des barres dans l'historique.

Déclaration

```
class COpenBuffer: public CDoubleBuffer
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

CHighBuffer

CHighBuffer est une classe permettant un accès simplifié aux prix les plus hauts des barres de l'historique.

Description

La classe CHighBuffer fournit un accès aux prix les plus hauts des barres de l'historique.

Déclaration

```
class CHighBuffer: public CDoubleBuffer
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

CLowBuffer

CCloseBuffer est une classe permettant un accès simplifié aux prix les plus bas des barres dans l'historique.

Description

La classe CHighBuffer fournit un accès aux prix les plus bas des barres de l'historique.

Déclaration

```
class CLowBuffer: public CDoubleBuffer
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

CCloseBuffer

CCloseBuffer est une classe permettant un accès simplifié aux prix de clôture des barres dans l'historique.

Description

La classe CCloseBuffer fournit un accès aux prix de clôture des barres de l'historique.

Déclaration

```
class CCloseBuffer: public CDoubleBuffer
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour le buffer
virtual RefreshCurrent	Met à jour la valeur courante

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour le buffer.

```
virtual bool  Refresh()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
virtual bool  RefreshCurrent()
```

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

CIndicatorBuffer

CIndicatorBuffer est une classe permettant un accès simplifié aux données du buffer de l'indicateur.

Description

La classe CIndicatorBuffer fournit un accès simplifié aux données du buffer de l'indicateur technique.

Déclaration

```
class CIndicatorBuffer: public CDoubleBuffer
```

Titre

```
#include <Indicators\Indicator.mqh>
```

Méthodes de Classe

Attributs	
Offset	Retourne/Définit le décalage du buffer
Name	Retourne/Définit le nom du buffer
Méthodes d'Accès aux Données	
At	Retourne l'élément du buffer
Méthodes de Mise à Jour des Données	
Refresh	Met à jour le buffer
RefreshCurrent	Met à jour seulement la valeur courante

© 2000-2015, MetaQuotes Software Corp.

Offset

Retourne le décalage du buffer

```
int  Offset() const
```

Valeur de retour

Décalage du buffer.

Offset

Définit le décalage du buffer.

```
void  Offset(
   const int  offset      // décalage
   )
```

Paramètres

offset

[in] Nouveau décalage du buffer.

© 2000-2015, MetaQuotes Software Corp.

Name

Retourne le nom du buffer.

```
string  Name() const
```

Valeur de retour

Nom du buffer.

Name

Définit le nom du buffer.

```
void  Name(
   const string  name      // nom
   )
```

Paramètres

name

[in] Nouveau nom du buffer.

© 2000-2015, MetaQuotes Software Corp.

At

Retourne l'élément du buffer par son index.

```
double  At(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

Element du buffer situé à l'index spécifié.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour le buffer en entier.

```
bool  Refresh(
   const int  handle,     // handle
   const int  num         // numéro du buffer
   )
```

Paramètres

handle

[in] Handle de l'indicateur.

num

[in] Index du buffer de l'indicateur.

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

RefreshCurrent

Met à jour l'élément courant (index 0) du buffer.

```
bool  RefreshCurrent(
   const int  handle,     // handle de l'indicateur
   const int  num         // numéro du buffer
   )
```

Paramètres

handle

[in] Handle de l'indicateur.

num

[in] Numéro du buffer.

Valeur de retour

vrai si réalisé avec succès, faux si le buffer n'a pas été mis à jour.

© 2000-2015, MetaQuotes Software Corp.

CSeries

CSeries est la classe de base pour accéder aux séries de données de la Bibliothèque Standard.

Description

La classe CSeries fournit l'accès aux fonctions des séries de données MQL5 à tous ses descendants.

Déclaration

```
class CSeries: public CArrayObj
```

Titre

```
#include <Indicators\Series.mqh>
```

Méthodes de Classe

Attributs	
Name	Retourne le nom de la série de données ou de l'indicateur
BuffersTotal	Retourne le nombre de buffers d'une série de données ou d'un indicateur
Timeframe	Retourne le flag de la période de la série de données ou de l'indicateur
Symbol	Retourne le symbole de la série de données ou de l'indicateur
Period	Retourne la période de la série de données ou de l'indicateur
RefreshCurrent	Retourne/Définit le flag de mise à jour des données courantes
Méthodes d'Accès aux Données	
virtual BufferResize	Définit la taille du buffer de la série de données ou de l'indicateur
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour les données de la série de données ou de l'indicateur
PeriodDescription	Retourne la période sous forme d'une chaîne de caractères

© 2000-2015, MetaQuotes Software Corp.

Name

Retourne le nom de la série de données ou de l'indicateur

```
string  Name() const
```

Valeur de retour

Le nom de la série de données ou de l'indicateur

© 2000-2015, MetaQuotes Software Corp.

BuffersTotal

Retourne le nombre de buffers d'une série de données ou d'un indicateur.

```
int  BuffersTotal() const
```

Valeur de retour

Le nombre de buffer de la série de données ou de l'indicateur.

Note

La série de données n'a qu'un seul buffer.

© 2000-2015, MetaQuotes Software Corp.

Timeframe

Retourne le flag de la période de la série de données ou de l'indicateur.

```
int  Timeframe() const
```

Valeur de retour

Le flag de la période de la série de données ou de l'indicateur.

Note

C'est le flag de visibilité de certaines périodes.

© 2000-2015, MetaQuotes Software Corp.

Symbol

Retourne le symbole de la série de données ou de l'indicateur.

```
string  Symbol() const
```

Valeur de retour

Le symbole de la série de données ou de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Period

Retourne la période de la série de données ou de l'indicateur.

```
ENUM_TIMEFRAMES  Period() const
```

Valeur de retour

La période (valeur de l'énumération ENUM_TIMEFRAMES) de la série de données ou de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

RefreshCurrent

Définit un flag pour mettre à jour les valeurs courantes de la série de données ou de l'indicateur.

```
string  RefreshCurrent(
   const bool  flag      // nouveau flag
   )
```

Paramètres

flag

[in] Nouveaux flag de visibilité.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

BufferSize

Retourne le nombre de données disponibles dans le buffer de la série de données ou de l'indicateur.

```
int  BufferSize() const
```

Valeur de retour

Nombre de données disponibles dans le buffer de la série de données ou de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

BufferResize

Définit la taille du buffer de la série de données ou de l'indicateur.

```
virtual bool  BufferResize(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

Tous les buffers des série de données ou de l'indicateur ont la même taille.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour les données de la série de données ou de l'indicateur.

```
virtual void  Refresh(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Périodes à mettre à jour (flag).

© 2000-2015, MetaQuotes Software Corp.

PeriodDescription

Retourne la représentation sous forme de chaîne de caractères de l'énumération ENUM_TIMEFRAMES spécifiée.

```
string  PeriodDescription(
   const int  val=0      // Valeur
   )
```

Paramètres

val=0

[in] Valeur à convertir.

Valeur de retour

La représentation sous forme de chaîne de caractères de l'énumération ENUM_TIMEFRAMES spécifiée.

Note

Si la valeur n'est pas spécifiée ou est égale à 0, retourne la période de la série de données ou de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

CPriceSeries

La classe CPriceSeries est la classe de base pour accéder aux données prix.

Description

La classe CSeries fournit un accès simplifié à tous ses descendants aux fonctions MQL5 permettant de travailler avec les prix.

Déclaration

```
class CPriceSeries: public CSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Création	
virtual BufferResize	Définit la taille du buffer.
Méthodes d'Accès aux Données	
virtual GetData	Retourne l'élément du buffer spécifié par son index.
Méthodes de Mise à Jour des Données	
virtual Refresh	Met à jour les séries de données
Méthodes de Recherche de Données	
virtual MinIndex	Retourne l'index du plus petit élément dans l'intervalle spécifié
virtual MinValue	Retourne la valeur et l'index de l'élément minimal du buffer spécifié dans un intervalle spécifié
virtual MaxIndex	Retourne l'index du plus grand élément dans l'intervalle spécifié
virtual MaxValue	Retourne la valeur et l'index de l'élément maximal du buffer spécifié dans un intervalle spécifié

© 2000-2015, MetaQuotes Software Corp.

BufferResize

Définit la nouvelle taille du buffer.

```
virtual void  BufferResize(
   const int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2015, MetaQuotes Software Corp.

GetData

Retourne l'élément du buffer spécifié par son index.

```
double  GetData(
   const int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément du buffer.

Valeur de retour

L'élément du buffer à l'index spécifié ou EMPTY_VALUE.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour les séries de données.

```
virtual void  Refresh(
   const int  flags=OBJ_ALL_PERIODS      // période flags
   )
```

Paramètres

flags=OBJ_ALL_PERIODS

[in] Périodes à mettre à jour (flag).

© 2000-2015, MetaQuotes Software Corp.

MinIndex

Retourne l'index du plus petit élément dans l'intervalle spécifié.

```
virtual int  MinIndex(
   const int  start,      // index de départ
   const int  count       // nombre d'éléments à analyser
   ) const
```

Paramètres

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

Valeur de retour

L'index de l'élément minimal dans l'intervalle donné, ou -1 en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

MinValue

Retourne la valeur et l'index de l'élément minimal dans un intervalle spécifié.

```
virtual double  MinValue(
   const int   start,     // index de départ
   const int   count,     // nombre d'éléments à analyser
   int&        index      // référence à une variable pour l'index
   ) const
```

Paramètres

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

index

[out] Référence à une variable de type integer.

Valeur de retour

La valeur de l'élément minimal du buffer dans un intervalle spécifié, ou EMPTY_VALUE en cas d'erreur.

Note

L'index du plus petit élément est stocké dans la variable index.

© 2000-2015, MetaQuotes Software Corp.

MaxIndex

Retourne l'index du plus grand élément dans l'intervalle spécifié.

```
virtual int  MaxIndex(
   const int  start,      // index de départ
   const int  count       // nombre d'éléments à analyser
   ) const
```

Paramètres

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

Valeur de retour

L'index de l'élément maximal du buffer spécifié dans l'intervalle donné, ou -1 en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

MaxValue

Retourne la valeur et l'index de l'élément maximal du buffer spécifié dans l'intervalle spécifié.

```
virtual double  MaxValue(
   const int    start,      // index de départ
   const int    count,      // nombre d'éléments à analyser
   int&         index       // référence à une variable pour l'index
   ) const
```

Paramètres

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

index

[out] Référence à une variable de type integer.

Valeur de retour

La valeur de l'élément maximal du buffer spécifié dans un intervalle spécifié, ou EMPTY_VALUE en cas d'erreur.

Note

L'index du plus grand élément est stocké dans la variable index.

© 2000-2015, MetaQuotes Software Corp.

CIndicator

CIndicator est la classe de base des classes d'indicateurs techniques de la Bibliothèque Standard MQL.

Description

La classe CIndicator fournit un accès simplifié et à tous ses descendants aux fonctions des indicateurs techniques de l'API MQL5.

Déclaration

```
class CIndicator: public CSeries
```

Titre

```
#include <Indicators\Indicator.mqh>
```

Méthodes de Classe

Attributs	
Handle	Retourne le handle de l'indicateur.
Status	Retourne le statut de l'indicateur.
FullRelease	Définit le mode de libération du handle de l'indicateur.
Création	
Create	Crée l'indicateurs
BufferResize	Définit la nouvelle taille du buffer
Méthodes d'Accès aux Données	
GetData	Copie les données depuis le buffer de l'indicateur
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données de l'indicateur
Recherche des Valeurs Min/Max	
Minimum	Retourne l'index de l'élément minimal du buffer spécifié dans un intervalle donné.
MinValue	Retourne la valeur et l'index de l'élément minimal du buffer spécifié dans un intervalle spécifié.
Maximum	Retourne l'index de l'élément maximal du buffer spécifié dans un intervalle donné.
MaxValue	Retourne la valeur et l'index de l'élément maximal du buffer spécifié dans un intervalle

© 2000-2015, MetaQuotes Software Corp.

	spécifié.
Conversion des Enumérations	
MethodDescription	Retourne la valeur de l'énumération ENUM_MA_METHOD sous forme de chaîne de caractères
PriceDescription	Retourne la valeur de l'énumération ENUM_APPLIED_PRICE sous forme de chaîne de caractères
VolumeDescription	Retourne la valeur de l'énumération ENUM_APPLIED_VOLUME sous forme de chaîne de caractères
Utilisation avec le graphique	
AddToChart	Ajoute l'indicateur au graphique
DeleteFromChart	Enlève l'indicateur du graphique

Classes dérivées :

- CiAC
- CiAD
- CiADX
- CiADXWilder
- CiAlligator
- CiAMA
- CiAO
- CiATR
- CiBands
- CiBearsPower
- CiBullsPower
- CiBWMFI
- CiCCI
- CiChaikin
- CiDEMA
- CiDeMarker
- CiEnvelopes
- CiForce
- CiFractals
- CiFrAMA
- CiGator
- CiIchimoku
- CiMA

© 2000-2015, MetaQuotes Software Corp.

- CiMACD
- CiMFI
- CiMomentum
- CiOBV
- CiOsMA
- CiRSI
- CiRVI
- CiSAR
- CiStdDev
- CiStochastic
- CiTEMA
- CiTriX
- CiVIDyA
- CiVolumes
- CiWPR

© 2000-2015, MetaQuotes Software Corp.

Handle

Retourne le handle de l'indicateur.

```
int  Handle() const
```

Valeur de retour

Handle de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Status

Retourne le statut de l'indicateur.

```
string  Status() const
```

Valeur de retour

Le statut de création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

FullRelease

Définit le mode de libération du handle de l'indicateur.

```
void  FullRelease(
   const bool  flag=true      // flag
   )
```

Paramètres

flag

[in] Nouvelle valeur du paramètre the handle release flag.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur with specified parameters.

```
bool  Create(
   const string           symbol,          // symbole
   const ENUM_TIMEFRAMES  period,          // période
   const ENUM_INDICATOR   type,            // type
   const int              num_params,      // nombre de paramètres
   const MqlParam&        params[]         // référence sur le tableau des paramètres
   )
```

Paramètres

symbol

[in] Nom du symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

type

[in] Type de l'indicateur (énumération ENUM_INDICATOR).

num_params

[in] Nombre de paramètres de l'indicateur.

params

[in] Référence sur le tableau de paramètres de l'indicateur.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

BufferResize

Définit la taille du buffer de l'indicateur.

```
virtual bool  BufferResize(
   const int  size      // Taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

Note

Tous les buffers de l'indicateur ont la même taille.

© 2000-2015, MetaQuotes Software Corp.

BarsCalculated

Retourne le nombre de données calculées pour l'indicateur.

```
int  BarsCalculated() const;
```

Valeur de Retour

Retourne le nombre de données calculées dans le buffer de l'indicateur, ou -1 en cas d'erreur (données pas encore calculées).

© 2000-2015, MetaQuotes Software Corp.

GetData

Retourne l'élément spécifié du buffer spécifié de l'indicateur. La méthode Refresh() devrait être appelée pour pouvoir travailler avec les données les plus récentes avant d'utiliser la méthode GetData.

```
double  GetData(
   const int  buffer_num,     // numéro du buffer
   const int  index           // index de l'élément
   ) const
```

Paramètres

buffer_num

[in] Numéro du buffer.

index

[in] Index de l'élément.

Valeur de retour

En cas de succès, retourne la valeur numérique de l'élément, ou EMPTY_VALUE en cas d'erreur.

GetData

Retourne les données du buffer de l'indicateur à partir d'une position donnée et le nombre de données à retourner.

```
int  GetData(
   const int      start_pos,      // position
   const int      count,          // nombre d'éléments demandés
   const int      buffer_num,     // numéro du buffer
   double&        buffer[]        // tableau destination pour les données
   ) const
```

Paramètres

start_pos

[in] Position de départ du buffer de l'indicateur.

count

[in] Nombre d'éléments à retourner.

buffer_num

[in] Numéro du buffer de l'indicateur.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

En cas de succès, retourne le nombre d'éléments reçus du buffer spécifié de l'indicateur, sinon -1.

GetData

© 2000-2015, MetaQuotes Software Corp.

Retourne les données du buffer de l'indicateur à partir d'une heure de départ et le nombre de données à retourner.

```
int  GetData(
   const datetime  start_time,     // date/heure de départ
   const int       count,          // nombre d'éléments à retourner
   const int       buffer_num,     // numéro du buffer
   double&         buffer[]        // tableau destination pour les données
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer_num

[in] Numéro du buffer de l'indicateur.

buffer

[in] Référence vers le tableau destination.

Valeur de retour

En cas de succès, retourne le nombre d'éléments reçus du buffer spécifié de l'indicateur, sinon -1.

GetData

Retourne les données du buffer de l'indicateur à partir d'une heure de départ et de fin et le nombre de données à retourner.

```
int  GetData(
   const datetime  start_time,     // date/heure de départ
   const datetime  stop_time,      // date/heure de fin
   const int       buffer_num,     // numéro du buffer
   double&         buffer[]        // tableau destination des données
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer_num

[in] Numéro du buffer de l'indicateur.

buffer

[in] Référence vers le tableau destination.

© 2000-2015, MetaQuotes Software Corp.

Valeur de retour

En cas de succès, retourne le nombre d'éléments reçus du buffer spécifié de l'indicateur, sinon -1.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour les données de l'indicateur. Il est recommendé d'appeler cette fonction avant d'utiliser GetData().

```
virtual void  Refresh(
   int  flags=OBJ_ALL_PERIODS      // flags
   )
```

Paramètres

flags=OBJ_ALL_PERIODS

[in] Flags des périodes à mettre à jour.

© 2000-2015, MetaQuotes Software Corp.

Minimum

Retourne l'index de l'élément minimal du buffer spécifié dans un intervalle donné.

```
int  Minimum(
   const int  buffer_num,      // numéro du buffer
   const int  start,           // index de départ
   const int  count            // nombre d'éléments à procéder
   ) const
```

Paramètres

buffer_num

[in] Numéro du buffer dans lequel la valeur doit être recherchée.

start

[in] Index de départ de la recherche.

count

[in] Nombre d'éléments à rechercher.

Valeur de retour

L'index de l'élément minimal du buffer spécifié dans un intervalle donné.

© 2000-2015, MetaQuotes Software Corp.

MinValue

Retourne la valeur et l'index de l'élément minimal du buffer spécifié dans un intervalle spécifié.

```
double  MinValue(
   const int   buffer_num,     // numéro du buffer
   const int   start,          // index de départ
   const int   count,          // nombre d'éléments à traiter
   int&        index           // référence
   ) const
```

Paramètres

buffer_num

[in] Numéro du buffer dans lequel la valeur doit être recherchée.

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

index

[out] Référence à une variable de type int pour l'index de l'élément minimal.

Valeur de retour

La valeur de l'élément minimal du buffer spécifié dans un intervalle spécifié.

Note

L'index du plus petit élément du buffer est stocké dans la variable index, passée par référence.

© 2000-2015, MetaQuotes Software Corp.

Maximum

Retourne l'index de l'élément maximal du buffer spécifié dans un intervalle donné.

```
int  Maximum(
   const int  buffer_num,     // numéro du buffer
   const int  start,          // index de départ
   const int  count           // nombre d'éléments à procéder
   ) const
```

Paramètres

buffer_num

[in] Numéro du buffer dans lequel la valeur doit être recherchée.

start

[in] Index de départ de la recherche.

count

[in] Nombre d'éléments à rechercher.

Valeur de retour

Index de l'élément maximal du buffer spécifié dans un intervalle donné.

© 2000-2015, MetaQuotes Software Corp.

MaxValue

Retourne la valeur et l'index de l'élément maximal du buffer spécifié dans un intervalle spécifié.

```
double  MaxValue(
   const int   buffer_num,     // numéro du buffer
   const int   start,          // index de départ
   const int   count,          // nombre d'éléments à traiter
   int&        index           // référence
   ) const
```

Paramètres

buffer_num

[in] Numéro du buffer dans lequel la valeur doit être recherchée.

start

[in] Index de départ.

count

[in] Nombre d'éléments à traiter.

index

[out] Référence à une variable de type int pour l'index de l'élément maximal.

Valeur de retour

La valeur de l'élément maximal du buffer spécifié dans un intervalle spécifié.

Note

L'index du plus grand élément du buffer est stocké dans la variable index, passée par référence.

© 2000-2015, MetaQuotes Software Corp.

MethodDescription

La fonction retourne la valeur de l'énumération ENUM_MA_METHOD sous forme d'une chaîne de caractères.

```
string  MethodDescription(
   const int  val      // Valeur
   ) const
```

Paramètres

val

[in] Valeur de l'énumération ENUM_MA_METHOD.

Valeur de retour

La valeur de l'énumération ENUM_MA_METHOD sous forme d'une chaîne de caractères.

© 2000-2015, MetaQuotes Software Corp.

PriceDescription

La fonction retourne la valeur de l'énumération ENUM_APPLIED_PRICE sous forme d'une chaîne de caractères.

```
string  PriceDescription(
   const int  val      // Valeur
   ) const
```

Paramètres

val

[in] Valeur de l'énumération ENUM_APPLIED_PRICE.

Valeur de retour

La valeur de l'énumération ENUM_APPLIED_PRICE sous forme d'une chaîne de caractères.

© 2000-2015, MetaQuotes Software Corp.

VolumeDescription

La fonction retourne la valeur de l'énumération ENUM_APPLIED_VOLUME sous forme d'une chaîne de caractères.

```
string  VolumeDescription(
   const int  val      // Valeur
   ) const
```

Paramètres

val

[in] Valeur de l'énumération ENUM_APPLIED_VOLUME.

Valeur de retour

La valeur de l'énumération ENUM_APPLIED_VOLUME sous forme d'une chaîne de caractères.

© 2000-2015, MetaQuotes Software Corp.

AddToChart

Ajoute l'indicateur au graphique.

```
bool  AddToChart(
   const long  chart,      // Identifiant du graphique
   const int  subwin       // sous-fenêtre du graphique
   )
```

Paramètres

chart

[in] Identifiant du graphique.

subwin

[in] Sous-fenêtre du graphique.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

DeleteFromChart

Enlève l'indicateur du graphique.

```
bool  DeleteFromChart(
   const long  chart,      // Identifiant du graphique
   const int  subwin       // sous-fenêtre du graphique
   )
```

Paramètres

chart

[in] Identifiant du graphique.

subwin

[in] Sous-fenêtre du graphique.

Valeur de retour

vrai si réalisé avec succès, faux en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

CIndicators

La classe CIndicators permmet d'avoir une liste d'instances de classes de séries de données et d'indicateurs techniques.

Description

La classe CIndicators fournit la création des instances de classes des indicateurs techniques, leur stockage et leur gestion (synchronisation des données, gestion du handle et de la mémoire).

Déclaration

```
class CIndicators: public CArrayObj
```

Titre

```
#include <Indicators\Indicators.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée l'indicateur technique
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données de tous les indicateurs techniques de la collection.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés.

```
CIndicator*  Create(
   const string           symbol,     // Symbole name
   const ENUM_TIMEFRAMES  period,     // Période
   const ENUM_INDICATOR   type,       // Type de l'indicateur
   const int              count,      // nombre de paramètres
   const MqlParam&        params      // Référence sur le tableau des paramètres
   )
```

Paramètres

symbol

[in] Nom du symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

type

[in] Type de l'indicateur (ENUM_INDICATOR).

count

[in] Nombre de paramètres de l'indicateur.

params

[in] Référence sur le tableau de paramètres de l'indicateur.

Valeur de retour

En cas de succès, retourne la référence sur l'indicateur créé, et NULL si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour les données de tous les indicateurs techniques de la collection.

```
int  Refresh()
```

Valeur de retour

Retourne les flags des périodes mises à jour.

© 2000-2015, MetaQuotes Software Corp.

Classes des séries de données

Cet ensemble de chapitres contient les détails techniques des classes des séries de données de la Bibliothèque Standard MQL5 et les descriptions de ses composants principaux.

Classe	Description
CiSpread	Fournit un accès aux données historiques des spreads
CiTime	Fournit un accès aux données historiques des heures d'ouverture des barres de l'historique
CiTickVolume	Fournit un accès aux données historiques des volumes des ticks des barres de l'historique
CiRealVolume	Fournit un accès aux données historiques des volumes réels des barres de l'historique
CiOpen	Fournit un accès aux données historiques des prix d'ouverture (open) des barres de l'historique
CiHigh	Fournit un accès aux données historiques des prix plus hauts (high) des barres de l'historique
CiLow	Fournit un accès aux données historiques des prix plus bas (low) des barres de l'historique
CiClose	Fournit un accès aux données historiques des prix de clôture (close) des barres de l'historique

© 2000-2015, MetaQuotes Software Corp.

CiSpread

CiSpread est une classe permettant d'accéder aux spreads des barres de l'historique.

Description

La classe CiSpread fournit l'accès aux spreads des barres de l'historique.

Déclaration

```
class CiSpread: public CSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
BufferResize	Définit la taille du buffer
Méthodes d'Accès aux Données	
GetData	Retourne la donnée
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données

© 2000-2015, MetaQuotes Software Corp.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux spreads de l'historique.

```
bool  Create(
   string           symbol,     // symbole
   ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2015, MetaQuotes Software Corp.

BufferResize

Définit la nouvelle taille de la série.

```
virtual void  BufferResize(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2015, MetaQuotes Software Corp.

GetData

Retourne l'élément de la série par son index.

```
int  GetData(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément à retourner.

Valeur de retour

L'élément du buffer de la série de données, ou 0.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,     // position de départ
   int   count,         // nombre d'éléments à récupérer
   int&  buffer         // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,     // heure de départ
   int       count,          // nombre d'éléments
   int&      buffer          // tableau destination
   ) const
```

Paramètres

start_time

© 2000-2015, MetaQuotes Software Corp.

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

```
int  GetData(
   datetime  start_time,     // heure de départ
   datetime  stop_time,      // heure de fin
   int&      buffer          // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour les données de la série de données.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Paramètres

flags

[in] Flags de la période.

© 2000-2015, MetaQuotes Software Corp.

CiTime

CiTime est une classe permettant d'accéder aux heures d'ouverture des barres de l'historique.

Description

La classe CiTime fournit l'accès aux prix d'ouverture des barres de l'historique.

Déclaration

```
class CiTime: public CSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
BufferResize	Définit la taille du buffer
Méthodes d'Accès aux Données	
GetData	Retourne la donnée
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données

© 2000-2015, MetaQuotes Software Corp.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux heures d'ouverture des barres de l'historique.

```
bool  Create(
   string           symbol,     // symbole
   ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2015, MetaQuotes Software Corp.

BufferResize

Définit la nouvelle taille de la série.

```
virtual void  BufferResize(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2015, MetaQuotes Software Corp.

GetData

Retourne l'élément de la série par son index.

```
datetime  GetData(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément à retourner.

Valeur de retour

L'élément du buffer de la série de données, ou 0.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,     // position de départ
   int   count,         // nombre d'éléments à récupérer
   long& buffer         // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,     // heure de départ
   int       count,          // nombre d'éléments
   long&     buffer          // tableau destination
   ) const
```

Paramètres

start_time

© 2000-2015, MetaQuotes Software Corp.

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

```
int  GetData(
   datetime  start_time,     // heure de départ
   datetime  stop_time,      // heure de fin
   long&     buffer          // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour les données de la série de données.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Paramètres

flags

[in] Flags de la période.

© 2000-2015, MetaQuotes Software Corp.

CiTickVolume

CiTickVolume est une classe permettant un accès simplifié aux volumes des ticks des barres de l'historique.

Description

La classe CiTickVolume fournit l'accès aux volumes des ticks des barres de l'historique.

Déclaration

```
class CiTickVolume: public CSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
BufferResize	Définit la taille du buffer
Méthodes d'Accès aux Données	
GetData	Retourne la donnée
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données

© 2000-2015, MetaQuotes Software Corp.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux volumes des ticks des barres de l'historique.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2015, MetaQuotes Software Corp.

BufferResize

Définit la nouvelle taille de la série.

```
virtual void  BufferResize(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2015, MetaQuotes Software Corp.

GetData

Retourne l'élément de la série par son index.

```
datetime  GetData(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément à retourner.

Valeur de retour

L'élément du buffer de la série de données, ou 0.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,     // position de départ
   int   count,         // nombre d'éléments à récupérer
   long& buffer         // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,     // heure de départ
   int       count,          // nombre d'éléments
   long&     buffer          // tableau destination
   ) const
```

Paramètres

start_time

© 2000-2015, MetaQuotes Software Corp.

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

```
int  GetData(
   datetime  start_time,     // heure de départ
   datetime  stop_time,      // heure de fin
   long&     buffer          // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour les données de la série de données.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Paramètres

flags

[in] Flags de la période.

© 2000-2015, MetaQuotes Software Corp.

CiRealVolume

CiRealVolume est une classe permettant un accès simplifié aux volumes rééls des barres de l'historique.

Description

La classe CiRealVolume fournit l'accès aux volumes rééls des barres de l'historique.

Déclaration

```
class CiRealVolume: public CSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
BufferResize	Définit la taille du buffer
Méthodes d'Accès aux Données	
GetData	Retourne la donnée
Méthodes de Mise à Jour des Données	
Refresh	Met à jour les données

© 2000-2015, MetaQuotes Software Corp.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux volumes réels des barres de l'historique.

```
bool  Create(
   string           symbol,     // symbole
   ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2015, MetaQuotes Software Corp.

BufferResize

Définit la nouvelle taille de la série.

```
virtual void  BufferResize(
   int  size      // nouvelle taille
   )
```

Paramètres

size

[in] Nouvelle taille du buffer.

© 2000-2015, MetaQuotes Software Corp.

GetData

Retourne l'élément de la série par son index.

```
datetime  GetData(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index de l'élément à retourner.

Valeur de retour

L'élément du buffer de la série de données, ou 0.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int   start_pos,     // position de départ
   int   count,         // nombre d'éléments à récupérer
   long& buffer         // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,     // heure de départ
   int       count,          // nombre d'éléments
   long&     buffer          // tableau destination
   ) const
```

Paramètres

start_time

© 2000-2015, MetaQuotes Software Corp.

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

```
int  GetData(
   datetime  start_time,     // heure de départ
   datetime  stop_time,      // heure de fin
   long&     buffer          // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour les données de la série de données.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Paramètres

flags

[in] Flags de la période.

© 2000-2015, MetaQuotes Software Corp.

CiOpen

CiOpen est une classe permettant d'accéder aux prix d'ouverture (open) des barres de l'historique.

Description

La classe CiOpen permet d'accéder aux prix d'ouverture (open) des barres de l'historique.

Déclaration

```
class CiOpen: public CPriceSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
Méthodes d'Accès aux Données	
GetData	Retourne la donnée

© 2000-2015, MetaQuotes Software Corp.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux prix d'ouverture (open) des barres de l'historique.

```
bool  Create(
   string           symbol,     // symbole
   ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2015, MetaQuotes Software Corp.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int    start_pos,      // position de départ
   int    count,          // nombre d'éléments à récupérer
   double& buffer         // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   double&     buffer         // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

© 2000-2015, MetaQuotes Software Corp.

```
int  GetData(
   datetime  start_time,     // heure de départ
   datetime  stop_time,      // heure de fin
   double&   buffer          // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

CiHigh

CiHigh est une classe permettant d'accéder aux prix plus hauts (high) des barres de l'historique.

Description

La classe CHigh fournit un accès aux prix les plus hauts des barres de l'historique.

Déclaration

```
class CiHigh: public CPriceSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
Méthodes d'Accès aux Données	
GetData	Retourne la donnée

© 2000-2015, MetaQuotes Software Corp.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux prix plus hauts (high) des barres de l'historique.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2015, MetaQuotes Software Corp.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int    start_pos,      // position de départ
   int    count,          // nombre d'éléments à récupérer
   double& buffer         // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   double&     buffer         // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

© 2000-2015, MetaQuotes Software Corp.

```
int  GetData(
   datetime  start_time,     // heure de départ
   datetime  stop_time,      // heure de fin
   double&   buffer          // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

CiLow

CiLow est une classe permettant d'accéder aux prix plus bas (low) des barres de l'historique.

Description

La classe CiLow fournit un accès aux prix les plus bas des barres de l'historique.

Déclaration

```
class CiLow: public CPriceSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
Méthodes d'Accès aux Données	
GetData	Retourne la donnée

© 2000-2015, MetaQuotes Software Corp.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux prix plus bas (low) des barres de l'historique.

```
bool  Create(
   string           symbol,     // symbole
   ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2015, MetaQuotes Software Corp.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int    start_pos,      // position de départ
   int    count,          // nombre d'éléments à récupérer
   double& buffer         // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   double&     buffer         // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

© 2000-2015, MetaQuotes Software Corp.

```
int  GetData(
   datetime  start_time,     // heure de départ
   datetime  stop_time,      // heure de fin
   double&   buffer          // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

CiClose

CiClose est une classe permettant d'accéder aux prix de clôture des barres de l'historique.

Description

La classe CiClose fournit un accès aux prix de clôture des barres de l'historique.

Déclaration

```
class CiClose: public CPriceSeries
```

Titre

```
#include <Indicators\TimeSeries.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée une série de données
Méthodes d'Accès aux Données	
GetData	Retourne la donnée

© 2000-2015, MetaQuotes Software Corp.

Create

Crée une série de données avec les paramètres spécifiés pour accéder aux prix de clôture des barres de l'historique.

```
bool  Create(
   string           symbol,     // symbole
   ENUM_TIMEFRAMES  period      // période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si la série de données n'a pas été créée.

© 2000-2015, MetaQuotes Software Corp.

GetData

Retourne l'élément d'une série de données par la position de départ et le nombre d'éléments.

```
int  GetData(
   int    start_pos,      // position de départ
   int    count,          // nombre d'éléments à récupérer
   double& buffer         // tableau destination
   ) const
```

Paramètres

start_pos

[in] Position de départ de la série de données.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence sur le tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et le nombre d'éléments.

```
int  GetData(
   datetime  start_time,      // heure de départ
   int       count,           // nombre d'éléments
   double&     buffer         // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

count

[in] Nombre d'éléments à retourner.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

GetData

Retourne l'élément d'une série de données par l'heure de départ et de fin.

© 2000-2015, MetaQuotes Software Corp.

```
int  GetData(
   datetime  start_time,     // heure de départ
   datetime  stop_time,      // heure de fin
   double&   buffer          // tableau destination
   ) const
```

Paramètres

start_time

[in] Date/heure de départ.

stop_time

[in] Date/heure de fin.

buffer

[in] Référence au tableau destination pour les données.

Valeur de retour

>=0 en cas de succcès, -1 en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

Classes d'Indicateurs de Tendance

Cet ensemble de chapitres contient les détails techniques des classes des indicateurs de Tendance de la Bibliothèque Standard MQL5 et les descriptions de ses composants principaux.

Classe/groupe	Description
CiADX	Average Directional Index
CiADXWilder	Average Directional Index par Welles Wilder
CiBands	Bandes de Bollinger®
CiEnvelopes	Enveloppes
CiIchimoku	Ichimoku Kinko Hyo
CiMA	Moyenne Mobile
CiSAR	Système Parabolic Stop And Reverse
CiStdDev	Standard Deviation
CiDEMA	Double Exponential Moving Average
CiTEMA	Triple Exponential Moving Average
CiFrAMA	Fractal Adaptive Moving Average
CiAMA	Adaptive Moving Average
CiVIDyA	Variable Index Dynamic Average

© 2000-2015, MetaQuotes Software Corp.

CiADX

La classe CiADX permet d'utiliser l'indicateur technique Average Directional Index.

Description

La classe CiADX fournit la création, la configuration et l'accès aux données de l'indicateur Average Directional Index.

Déclaration

```
class CiADX: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer de la ligne principale
Plus	Retourne l'élément du buffer de la ligne +DI
Minus	Retourne l'élément du buffer de la ligne -DI
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Plus

Retourne l'élément du buffer de la ligne +DI de l'index spécifié.

```
double   Plus(
   int   index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne +DI de l'index spécifié, ou ***<t3>EMPTY_VALUE s'il n'y a aucune donnée correcte.</t3>

© 2000-2015, MetaQuotes Software Corp.

Minus

Retourne l'élément du buffer de la ligne -DI de l'index spécifié.

```
double  Minus(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne -DI de l'index spécifié, ou EMPTY_VALUE s'il n'y a aucune donnée correcte.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ADX for CiADX).

© 2000-2015, MetaQuotes Software Corp.

CiADXWilder

La classe CiADX permet d'utiliser l'indicateur technique Average Directional Index de Welles Wilder.

Description

La classe CiADXWilder fournit la création, la configuration et l'accès aux données de l'indicateur Average Directional Index de Welles Wilder.

Déclaration

```
class CiADXWilder: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer de la ligne principale
Plus	Retourne l'élément du buffer de la ligne +DI
Minus	Retourne l'élément du buffer de la ligne -DI
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Plus

Retourne l'élément du buffer de la ligne +DI de l'index spécifié.

```
double   Plus(
   int   index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne +DI de l'index spécifié, ou ***<t3>EMPTY_VALUE s'il n'y a aucune donnée correcte.</t3>

© 2000-2015, MetaQuotes Software Corp.

Minus

Retourne l'élément du buffer de la ligne -DI de l'index spécifié.

```
double  Minus(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne -DI de l'index spécifié, ou EMPTY_VALUE s'il n'y a aucune donnée correcte.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ADXW for CiADXWilder).

© 2000-2015, MetaQuotes Software Corp.

CiBands

La classe CiBands permet d'utiliser l'indicateur technique des Bandes de Bollinger®.

Description

La classe CiBands fournit la création, la configuration et l'accès aux données de l'indicateur des Bandes de Bollinger.

Déclaration

```
class CiBands: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
MaShift	Retourne le décalage horizontal
Deviation	Retourne la déviation
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer de la ligne de base
Upper	Retourne l'élément du buffer de la ligne supérieure
Lower	Retourne l'élément du buffer de la ligne inférieure
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaShift

Retourne le décalage horizontal de l'indicateur.

```
int  MaShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Deviation

Retourne la déviation.

```
double  Deviation() const
```

Valeur de retour

Retourne la déviation définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage
   double           deviation,     // Déviation
   int              applied        // prix à appliquer ou handle
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal de l'indicateur.

deviation

[in] Déviation.

applied

[in] Type de volume à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Base

Retourne l'élément du buffer de la ligne de base de l'index spécifié.

```
double  Base(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne de Base de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Upper

Retourne l'élément du buffer de la ligne supérieure de l'index spécifié.

```
double  Upper(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne supérieure de l'index spécifié, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Lower

Retourne l'élément du buffer de la ligne inférieure de l'index spécifié.

```
double  Lower(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne inférieure de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_BANDS for CiBands).

© 2000-2015, MetaQuotes Software Corp.

CiEnvelopes

CiEnvelopes est la classe permettant d'utiliser l'indicateur technique Enveloppes.

Description

La classe CiEnvelopes fournit la création, la configuration et l'accès aux données de l'indicateur Enveloppes.

Déclaration

```
class CiEnvelopes: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
MaShift	Retourne le décalage horizontal
MaMethod	Retourne la méthode de lissage
Deviation	Retourne la déviation
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Upper	Retourne l'élément du buffer de la ligne supérieure
Lower	Retourne l'élément du buffer de la ligne inférieure
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaShift

Retourne le décalage horizontal de l'indicateur.

```
int  MaShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Deviation

Retourne la valeur de la déviation.

```
double  Deviation() const
```

Valeur de retour

Retourne la valeur de déviation définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage horizontal
   ENUM_MA_METHOD   ma_method,     // Méthode de lissage
   int              applied,       // prix type ou handle to apply
   double           deviation      // Déviation
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Prix type of handle to apply.

deviation

[in] Déviation.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Upper

Retourne l'élément du buffer de la ligne supérieure de l'index spécifié.

```
double  Upper(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne supérieure de l'index spécifié, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Lower

Retourne l'élément du buffer de la ligne inférieure de l'index spécifié.

```
double  Lower(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne inférieure de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ENVELOPES for CiEnvelopes).

© 2000-2015, MetaQuotes Software Corp.

CiIchimoku

La classe CiIchimoku permet d'utiliser l'indicateur technique Ichimoku Kinko Hyoe.

Description

La classe CiIchimoku fournit la création, la configuration et l'accès aux données de l'indicateur Ichimoku Kinko Hyo.

Déclaration

```
class CiIchimoku: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
TenkanSenPeriod	Retourne la période TenkanSen
KijunSenPeriod	Retourne la période KijunSen
SenkouSpanBPeriod	Retourne la période SenkouSpanB
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
TenkanSen	Retourne l'élément du buffer de la ligne TenkanSen
KijunSen	Retourne l'élément du buffer de la ligne KijunSen
SenkouSpanA	Retourne l'élément du buffer de la ligne SenkouSpanA
SenkouSpanB	Retourne l'élément du buffer de la ligne SenkouSpanB
ChinkouSpan	Retourne l'élément du buffer de la ligne ChikouSpan
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

TenkanSenPeriod

Retourne la période TenkanSen.

```
int  TenkanSenPeriod() const
```

Valeur de retour

Retourne la période TenkanSen, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

KijunSenPeriod

Retourne la période KijunSen

```
int  KijunSenPeriod() const
```

Valeur de retour

Retourne la période KijunSen, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

SenkouSpanBPeriod

Retourne la période SenkouSpanB.

```
int  SenkouSpanBPeriod() const
```

Valeur de retour

Retourne la période SenkouSpanB, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,           // Symbole
   ENUM_TIMEFRAMES  period,           // Période
   int              tenkan_sen,       // Période of TenkanSen
   int              kijun_sen,        // Période of KijunSen
   int              senkou_span_b     // Période of SenkouSpanB
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

tenkan_sen

[in] Période de TenkanSen.

kijun_sen

[in] Période de KijunSen.

senkou_span_b

[in] Période de SenkouSpanB.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

TenkanSen

Retourne l'élément du buffer de la ligne TenkanSen de l'index spécifié.

```
double  TenkanSen(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne TenkanSen de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

KijunSen

Retourne l'élément du buffer de la ligne KijunSen de l'index spécifié.

```
double  KijunSen(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne KijunSen de l'index spécifié, ou ***<t3>EMPTY_VALUE s'il n'y a aucune donnée correcte.</t3>

© 2000-2015, MetaQuotes Software Corp.

SenkouSpanA

Retourne l'élément du buffer de la ligne SenkouSpanA de l'index spécifié.

```
double   SenkouSpanA(
   int   index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne SenkouSpanA de l'index spécifié, ou ***<t3>EMPTY_VALUE s'il n'y a aucune donnée correcte.</t3>

© 2000-2015, MetaQuotes Software Corp.

SenkouSpanB

Retourne l'élément du buffer de la ligne SenkouSpanB de l'index spécifié.

```
double   SenkouSpanB(
   int   index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne SenkouSpanB de l'index spécifié, ou ***<t3>EMPTY_VALUE s'il n'y a aucune donnée correcte.</t3>

© 2000-2015, MetaQuotes Software Corp.

ChinkouSpan

Retourne l'élément du buffer de la ligne ChinkouSpan de l'index spécifié.

```
double  ChinkouSpan(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne ChinkouSpan de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ICHIMOKU for Cilchimoku).

© 2000-2015, MetaQuotes Software Corp.

CiMA

La classe CiMA permet d'utiliser l'indicateur technique Moving Average (Moyenne Mobile).

Description

La classe CiMA fournit la création, la configuration et l'accès aux données de l'indicateur Moving Average (Moyenne Mobile).

Déclaration

```
class CiMA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
MaShift	Retourne le décalage horizontal
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaShift

Retourne le décalage horizontal de l'indicateur.

```
int  MaShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage (valeur de l'énumération ENUM_MA_METHOD), définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           string,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage horizontal
   ENUM_MA_METHOD   ma_method,     // Méthode de lissage
   int              applied        // prix type of handle to apply
   )
```

Paramètres

string

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_MA for CiMA).

© 2000-2015, MetaQuotes Software Corp.

CiSAR

La classe CiSAR permet d'utiliser l'indicateur technique Parabolic Stop And Reverse System.

Description

La classe CiSAR fournit la création, la configuration et l'accès aux données de l'indicateur Parabolic Stop And Reverse System.

Déclaration

```
class CiSAR: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
SarStep	Retourne le pas d'accroissement de la vélocité
Maximum	Retourne le coefficient de suivi du prix
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

SarStep

Retourne le pas d'accroissement de la vélocité (coefficient d'accélération).

```
double  SarStep() const
```

Valeur de retour

Le pas d'accroissement de la vélocité, défini à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Maximum

Retourne le coefficient de suivi du prix.

```
double  Maximum() const
```

Valeur de retour

Le coefficient de suivi du prix, défini à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,      // Symbole
   ENUM_TIMEFRAMES  period,      // Période
   double           step,        // Pas
   double           maximum      // Coefficient
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

step

[in] Pas d'accroissement de la vélocité.

maximum

[in] Prix following coefficient.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_SAR for CiSAR).

© 2000-2015, MetaQuotes Software Corp.

CiStdDev

La classe CiStdDev permet d'utiliser l'indicateur technique Standard Deviation.

Description

La classe CiStdDev fournit la création, la configuration et l'accès aux données de l'indicateur technique Standard Deviation.

Déclaration

```
class CiStdDev: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
MaShift	Retourne le décalage horizontal
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaShift

Retourne le décalage horizontal de l'indicateur.

```
int  MaShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage (valeur de l'énumération ***<t1> </t1><li2>ENUM_MA_METHOD</li2><t4>), définie à la création de l'indicateur.</t4>

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage horizontal
   ENUM_MA_METHOD   ma_method,     // Méthode de lissage
   int              applied        // type de prix ou handle à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_STDDEV for CiStdDev).

© 2000-2015, MetaQuotes Software Corp.

CiDEMA

La classe CiDEMA permet d'utiliser l'indicateur technique Double Exponential Moving Average.

Description

La classe CiDEMA fournit la création, la configuration et l'accès aux données de l'indicateur Double Exponential Moving Average.

Déclaration

```
class CiDEMA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
IndShift	Retourne le décalage horizontal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

IndShift

Retourne le décalage horizontal de l'indicateur.

```
int  IndShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           string,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ind_shift,     // Décalage
   int              applied        // prix type of handle to apply
   )
```

Paramètres

string

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ind_shift

[in] Décalage horizontal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_DEMA for CiDEMA).

© 2000-2015, MetaQuotes Software Corp.

CiTEMA

La classe CiTEMA permet d'utiliser l'indicateur technique Triple Exponential Moving Average.

Description

La classe CiTEMA fournit la création, la configuration et l'accès aux données de l'indicateur Triple Exponential Moving Average.

Déclaration

```
class CiTEMA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
IndShift	Retourne le décalage horizontal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

IndShift

Retourne le décalage horizontal de l'indicateur.

```
int  IndShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage
   int              applied        // prix type of handle to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_TEMA for CiTEMA).

© 2000-2015, MetaQuotes Software Corp.

CiFrAMA

La classe CiFrAMA permet d'utiliser l'indicateur technique Fractal Adaptive Moving Average.

Description

La classe CiFrAMA fournit la création, la configuration et l'accès aux données de l'indicateur Fractal Adaptive Moving Average.

Déclaration

```
class CiFrAMA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
IndShift	Retourne le décalage horizontal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

IndShift

Retourne le décalage horizontal de l'indicateur.

```
int  IndShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              ma_shift,      // Décalage
   int              applied        // prix type of handle to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_shift

[in] Décalage horizontal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_FRAMA for CiFrAMA).

© 2000-2015, MetaQuotes Software Corp.

CiAMA

La classe CiTriX permet d'utiliser l'indicateur technique Adaptive Moving Average.

Description

La classe CiAMA fournit la création, la configuration et l'accès aux données de l'indicateur Adaptive Moving Average.

Déclaration

```
class CiAMA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
FastEmaPeriod	Retourne la période de lissage de la moyenne mobile rapide.
SlowEmaPeriod	Retourne la période de lissage de la moyenne mobile lente.
IndShift	Retourne le décalage horizontal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

FastEmaPeriod

Retourne la période de lissage de la moyenne mobile rapide.

```
int  FastEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile rapide, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

SlowEmaPeriod

Retourne la période de lissage de la moyenne mobile lente.

```
int  SlowEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile lente, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

IndShift

Retourne le décalage horizontal de l'indicateur.

```
int  IndShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           string,              // Symbole
   ENUM_TIMEFRAMES  period,              // Période
   int              ma_period,           // Période de lissage
   int              fast_ema_period,     // Période de la moyenne mobile rapide
   int              slow_ema_period,     // Période de la moyenne mobile lente
   int              ind_shift,           // Décalage
   int              applied              // prix type ou handle to apply
   )
```

Paramètres

string

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

fast_ema_period

[in] Période de la moyenne mobile rapide.

slow_ema_period

[in] Période de la moyenne mobile lente.

ind_shift

[in] Décalage horizontal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_AMA for CiAMA).

© 2000-2015, MetaQuotes Software Corp.

CiVIDyA

La classe CiVIDyA permet d'utiliser l'indicateur technique Variable Index DYnamic Average.

Description

La classe CiVIDyA fournit la création, la configuration et l'accès aux données de l'indicateur ariable Index DYnamic Average.

Déclaration

```
class CiVIDyA: public CIndicator
```

Titre

```
#include <Indicators\Trend.mqh>
```

Méthodes de Classe

Attributs	
CmoPeriod	Retourne la période du Momentum
EmaPeriod	Retourne la période de lissage de la moyenne mobile.
IndShift	Retourne le décalage horizontal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

CmoPeriod

Retourne la période du Momentum.

```
int  CmoPeriod() const
```

Valeur de retour

Retourne la période du Momentum, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

EmaPeriod

Retourne la période de lissage de la moyenne mobile.

```
int  EmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

IndShift

Retourne le décalage horizontal de l'indicateur.

```
int  IndShift() const
```

Valeur de retour

Retourne la valeur du décalage horizontal définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,          // Symbole
   ENUM_TIMEFRAMES  period,          // Période
   int              cmo_period,      // Période du Momentum
   int              ema_period,      // Période de lissage
   int              ind_shift,       // Décalage
   int              applied          // prix type ou handle to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

cmo_period

[in] Période du Momentum.

ema_period

[in] Période de lissage.

ind_shift

[in] Décalage horizontal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_VIDYA for CiVIDyA).

© 2000-2015, MetaQuotes Software Corp.

Oscillateurs

Cet ensemble de chapitres contient les détails techniques des classes d'Oscillateurs de la Bibliothèque Standard MQL5 et les descriptions de ses composants principaux.

Classe/groupe	Description
CiATR	Average True Range
CiBearsPower	Bears Power
CiBullsPower	Bulls Power
CiCCI	Commodity Channel Index
CiChaikin	Chaikin Oscillator
CiDeMarker	DeMarker
CiForce	Force Index
CiMACD	Moving Averages Convergence-Divergence
CiMomentum	Momentum
CiOsMA	Moyenne Mobile de l'Oscillateur (Histogramme du MACD)
CiRSI	Relative Strength Index
CiRVI	Relative Vigor Index
CiStochastic	Stochastic Oscillator
CiWPR	Williams' Percent Range
CiTriX	Triple Exponential Moving Averages Oscillator

© 2000-2015, MetaQuotes Software Corp.

CiATR

La classe CiATR permet d'utiliser l'indicateur technique Average True Range.

Description

La classe CiATR fournit la création, la configuration et l'accès aux données de l'indicateur Average True Range.

Déclaration

```
class CiATR: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ATR for CiATR).

© 2000-2015, MetaQuotes Software Corp.

CiBearsPower

La classe CiBearsPower permet d'utiliser l'indicateur technique Bears Power (Pouvoir des Baissiers).

Description

La classe CiBearsPower fournit la création, la configuration et l'accès aux données de l'indicateur Bears Power (Puissance des Baissiers).

Déclaration

```
class CiBearsPower: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_BEARS for CiBearsPower).

© 2000-2015, MetaQuotes Software Corp.

CiBullsPower

La classe CiBullsPower permet d'utiliser l'indicateur technique Bulls Power (Pouvoir des Haussiers).

Description

La classe CiBullsPower fournit la création, la configuration et l'accès aux données de l'indicateur Bulls Power (Puissance des Haussiers).

Déclaration

```
class CiBullsPower: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_BULLS for CiBullsPower).

© 2000-2015, MetaQuotes Software Corp.

CiCCI

La classe CiCCI permet d'utiliser l'indicateur technique Commodity Channel Index.

Description

La classe CiCCI fournit la création, la configuration et l'accès aux données de l'indicateur Commodity Channel Index.

Déclaration

```
class CiCCI: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              applied        // type de prix ou handle à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_CCI for CiCCI).

© 2000-2015, MetaQuotes Software Corp.

CiChaikin

La classe CiChaikin permet d'utiliser l'indicateur technique Chaikin Oscillator.

Description

La classe CiChaikin fournit la création, la configuration et l'accès aux données de l'indicateur Chaikin Oscillator.

Déclaration

```
class CiChaikin: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
FastMaPeriod	Retourne la période de lissage de la moyenne mobile rapide
SlowMaPeriod	Retourne la période de lissage de la moyenne mobile lente
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

FastMaPeriod

Retourne la période de lissage de la moyenne mobile rapide.

```
int  FastMaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile rapide, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

SlowMaPeriod

Retourne la période de lissage de la moyenne mobile lente.

```
int  SlowMaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile lente, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string               symbol,             // Symbole
   ENUM_TIMEFRAMES      period,             // Période
   int                  fast_ma_period,     // Période de la moyenne mobile rapide
   int                  slow_ma_period,     // Période de la moyenne mobile lente
   ENUM_MA_METHOD       ma_method,          // Méthode de lissage
   ENUM_APPLIED_VOLUME  applied             // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

fast_ma_period

[in] Période de la moyenne mobile rapide.

slow_ma_period

[in] Période de la moyenne mobile lente.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_CHAIKIN for CiChaikin).

© 2000-2015, MetaQuotes Software Corp.

CiDeMarker

CiDeMarker est une classe permettant d'utiliser l'indicateur technique DeMarker.

Description

La classe CiDeMarker fournit la création, la configuration et l'accès aux données de l'indicateur DeMarker.

Déclaration

```
class CiDeMarker: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_DEMARKER for CiDeMarker).

© 2000-2015, MetaQuotes Software Corp.

CiForce

La classe CiForce permet d'utiliser l'indicateur technique Force Index.

Description

La classe CiForce fournit la création, la configuration et l'accès aux données de l'indicateur Force Index.

Déclaration

```
class CiForce: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string               symbol,        // symbole
   ENUM_TIMEFRAMES      period,        // Période
   int                  ma_period,     // Période de lissage
   ENUM_MA_METHOD       ma_method,     // Méthode de lissage
   ENUM_APPLIED_VOLUME  applied        // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_FORCE for CiForce).

© 2000-2015, MetaQuotes Software Corp.

CiMACD

La classe CiMACD permet d'utiliser l'indicateur technique Moving Averages Convergence-Divergence.

Description

La classe CiMACD fournit la création, la configuration et l'accès aux données de l'indicateur Moving Averages Convergence-Divergence.

Déclaration

```
class CiMACD: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
FastEmaPeriod	Retourne la période de lissage de la moyenne mobile rapide.
SlowEmaPeriod	Retourne la période de lissage de la moyenne mobile lente.
SignalPeriod	Retourne la période de lissage de la ligne de signal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer de la ligne principale
Signal	Retourne l'élément du buffer de la ligne de signal
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

FastEmaPeriod

Retourne la période de lissage de la moyenne mobile rapide.

```
int  FastEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile rapide, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

SlowEmaPeriod

Retourne la période de lissage de la moyenne mobile lente.

```
int  SlowEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile lente, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

SignalPeriod

Retourne la période de lissage de la ligne de signal.

```
int  SignalPeriod() const
```

Valeur de retour

Retourne la période de lissage de la ligne de signal définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,              // symbole
   ENUM_TIMEFRAMES  period,              // Période
   int              fast_ema_period,     // Période de la moyenne mobile rapide
   int              slow_ema_period,     // Période de la moyenne mobile lente
   int              signal_period,       // Période du signal
   int              applied              // prix type ou handle to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

fast_ema_period

[in] Période de la moyenne mobile rapide.

slow_ema_period

[in] Période de la moyenne mobile lente.

signal_period

[in] Période de la ligne de signal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Signal

Retourne l'élément du buffer de la ligne de signal de l'index spécifié.

```
double  Signal(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne de signal à l'index spécifié, ou EMPTY_VALUE si aucune donnée correcte n'est présente.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_MACD for CiMACD).

© 2000-2015, MetaQuotes Software Corp.

CiMomentum

CiMomentum est une classe permettant d'utiliser l'indicateur technique Momentum.

Description

La classe CiMomentum fournit la création, la configuration et l'accès aux données de l'indicateur Momentum.

Déclaration

```
class CiMomentum: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              applied        // type de prix ou handle à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_MOMENTUM for CiMomentum).

© 2000-2015, MetaQuotes Software Corp.

CiOsMA

La classe CiOsMA permet d'utiliser l'indicateur technique Moving Average of Oscillator (histogramme du MACD).

Description

La classe CiOsMA fournit la création, la configuration et l'accès aux données de l'indicateur Moving Average of Oscillator (histogramme de MACD).

Déclaration

```
class CiOsMA: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
FastEmaPeriod	Retourne la période de lissage de la moyenne mobile rapide.
SlowEmaPeriod	Retourne la période de lissage de la moyenne mobile lente.
SignalPeriod	Retourne la période de lissage de la ligne de signal
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

FastEmaPeriod

Retourne la période de lissage de la moyenne mobile rapide.

```
int  FastEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile rapide, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

SlowEmaPeriod

Retourne la période de lissage de la moyenne mobile lente.

```
int  SlowEmaPeriod() const
```

Valeur de retour

Retourne la période de lissage de la moyenne mobile lente, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

SignalPeriod

Retourne la période de lissage de la ligne de signal.

```
int  SignalPeriod() const
```

Valeur de retour

Retourne la période de lissage de la ligne de signal définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,              // symbole
   ENUM_TIMEFRAMES  period,              // Période
   int              fast_ema_period,     // Période de la moyenne mobile rapide
   int              slow_ema_period,     // Période de la moyenne mobile lente
   int              signal_period,       // Période de la ligne de signal
   int              applied              // prix type ou handle to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

fast_ema_period

[in] Période de la moyenne mobile rapide.

slow_ema_period

[in] Période de la moyenne mobile lente.

signal_period

[in] Période de la ligne de signal.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_OSMA for CiOsMA).

© 2000-2015, MetaQuotes Software Corp.

CiRSI

La classe CiRSI permet d'utiliser l'indicateur technique Relative Strength Index.

Description

La classe CiRSI fournit la création, la configuration et l'accès aux données de l'indicateur Relative Strength Index.

Déclaration

```
class CiRSI: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              applied        // type de prix ou handle à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

applied

[in] Type du prix ou handle à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_RSI for CiRSI).

© 2000-2015, MetaQuotes Software Corp.

CiRVI

La classe CiRVI permet d'utiliser l'indicateur technique Relative Vigor Index.

Description

La classe CiRVI fournit la création, la configuration et l'accès aux données de l'indicateur Relative Vigor Index.

Déclaration

```
class CiRVI: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer de la ligne de base
Signal	Retourne l'élément du buffer de la ligne de signal
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period      // Période de lissage
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Signal

Retourne l'élément du buffer de la ligne de signal de l'index spécifié.

```
double  Signal(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne de signal à l'index spécifié, ou EMPTY_VALUE si aucune donnée correcte n'est présente.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_RVI for CiRVI).

© 2000-2015, MetaQuotes Software Corp.

CiStochastic

La classe CiChaikin permet d'utiliser l'indicateur technique Stochastique.

Description

La classe CiStochastic fournit la création, la configuration et l'accès aux données de l'indicateur Stochastique.

Déclaration

```
class CiStochastic: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
Kperiod	Retourne la période de lissage de la ligne %K
Dperiod	Retourne la période de lissage de la ligne %D
Slowing	Retourne la période de ralentissement
MaMethod	Retourne la méthode de lissage
PriceField	Type de prix (Low/High ou Close/Close) à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer de la ligne de base
Signal	Retourne l'élément du buffer de la ligne de signal
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

Kperiod

Retourne la période de lissage de la ligne %K.

```
int  Kperiod() const
```

Valeur de retour

Retourne la période de lissage de la ligne %K définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Dperiod

Retourne la période de lissage de la ligne %D.

```
int  Dperiod() const
```

Valeur de retour

Retourne la période de lissage de la ligne %D définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Slowing

Retourne la période de ralentissement.

```
int  Slowing() const
```

Valeur de retour

Retourne la période de ralentissement définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

PriceField

Retourne le prix type (Low/High ou Close/Close) to apply.

```
ENUM_STO_PRICE  PriceField() const
```

Valeur de retour

x Le prix type xprice type to apply, defined at the indicator creation.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,          // symbole
   ENUM_TIMEFRAMES  period,          // Période
   int              Kperiod,         // Période de lissage de %K
   int              Dperiod,         // Période de lissage de %D
   int              slowing,         // Période de ralentissement
   ENUM_MA_METHOD   ma_method,       // Méthode de lissage
   ENUM_STO_PRICE   price_field      // prix type to apply
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Kperiod

[in] Période de lissage de la ligne %K.

Dperiod

[in] Période de lissage de la ligne %D.

slowing

[in] Période de ralentissement.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

price_field

[in] Prix type (Low/High ou Close/Close) to apply (ENUM_STO_PRICE enumeration).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Signal

Retourne l'élément du buffer de la ligne de signal de l'index spécifié.

```
double  Signal(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Signal

Retourne l'élément du buffer de la ligne de signal de l'index spécifié.

```
double  Signal(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne de signal à l'index spécifié, ou EMPTY_VALUE si aucune donnée correcte n'est présente.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_STOCHASTIC for CiStochastic).

© 2000-2015, MetaQuotes Software Corp.

CiTriX

La classe CiTriX permet d'utiliser l'indicateur technique Triple Exponential Moving Averages Oscillator.

Description

La classe CiTriX fournit la création, la configuration et l'accès aux données de l'indicateur Triple Exponential Moving Averages Oscillator.

Déclaration

```
class CiTriX: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,        // symbole
   ENUM_TIMEFRAMES  period,        // Période
   int              ma_period,     // Période de lissage
   int              applied        // prix type ou handle
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

applied

[in] Prix type of handle to apply.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_TRIX for CiTriX).

© 2000-2015, MetaQuotes Software Corp.

CiWPR

La classe CiWPR permet d'utiliser l'indicateur technique Williams' Percent Range.

Description

La classe CiWPR fournit la création, la configuration et l'accès aux données de l'indicateur Williams' Percent Range.

Déclaration

```
class CiWPR: public CIndicator
```

Titre

```
#include <Indicators\Oscilators.mqh>
```

Méthodes de Classe

Attributs	
CalcPeriod	Retourne la période de calcul
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

CalcPeriod

Retourne la période de calcul.

```
int  CalcPeriod() const
```

Valeur de retour

Retourne la période de calcul définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,          // symbole
   ENUM_TIMEFRAMES  period,          // Période
   int              calc_period      // Période de calcul
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

calc_period

[in] Période de calcul.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_WPR for CiWPR).

© 2000-2015, MetaQuotes Software Corp.

Indicateurs de Volumes

Cet ensemble de chapitres contient les détails techniques des classes des indicateurs de Volume de la Bibliothèque Standard MQL5 et les descriptions de ses composants principaux.

Classe/groupe	Description
CiAD	Accumulation/Distribution
CiMFI	Money Flow Index
CiOBV	On Balance Volume
CiVolumes	Volumes

© 2000-2015, MetaQuotes Software Corp.

CiAD

CiAD est une classe permettant d'utiliser l'indicateur technique Accumulation/Distribution.

Description

La classe CiAD fournit la création, la configuration et l'accès aux données de l'indicateur Accumulation/Distribution.

Déclaration

```
class CiAD: public CIndicator
```

Titre

```
#include <Indicators\Volumes.mqh>
```

Méthodes de Classe

Attributs	
Applied	Retourne le type de volume à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string               symbol,      // symbole
   ENUM_TIMEFRAMES      period,      // Période
   ENUM_APPLIED_VOLUME  applied      // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_AD for CiAD).

© 2000-2015, MetaQuotes Software Corp.

CiMFI

La classe CiMFI permet d'utiliser l'indicateur technique Money Flow Index.

Description

La classe CiMFI fournit la création, la configuration et l'accès aux données de l'indicateur Money Flow Index.

Déclaration

```
class CiMFI: public CIndicator
```

Titre

```
#include <Indicators\Volumes.mqh>
```

Méthodes de Classe

Attributs	
MaPeriod	Retourne la période de lissage
Applied	Retourne le type de volume à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

MaPeriod

Retourne la période de lissage.

```
int  MaPeriod() const
```

Valeur de retour

Retourne la période de lissage définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string               symbol,        // symbole
   ENUM_TIMEFRAMES      period,        // Période
   int                  ma_period,     // Période de lissage
   ENUM_APPLIED_VOLUME  applied        // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

ma_period

[in] Période de lissage.

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_MFI for CiMFI).

© 2000-2015, MetaQuotes Software Corp.

CiOBV

La classe CiOBV permet d'utiliser l'indicateur technique On Balance Volume.

Description

La classe CiOBV fournit la création, la configuration et l'accès aux données de l'indicateur On Balance Volume.

Déclaration

```
class CiOBV: public CIndicator
```

Titre

```
#include <Indicators\Volumes.mqh>
```

Méthodes de Classe

Attributs	
Applied	Retourne le type de volume à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string               symbol,      // symbole
   ENUM_TIMEFRAMES      period,      // Période
   ENUM_APPLIED_VOLUME  applied      // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_OBV for CiOBV).

© 2000-2015, MetaQuotes Software Corp.

CiVolumes

La classe CiVolumes permet d'utiliser l'indicateur technique des Volumes.

Description

La classe CiVolumes fournit la création, la configuration et l'accès aux données de l'indicateur Volumes.

Déclaration

```
class CiVolumes: public CIndicator
```

Titre

```
#include <Indicators\Volumes.mqh>
```

Méthodes de Classe

Attributs	
Applied	Retourne le type de volume à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string               symbol,      // symbole
   ENUM_TIMEFRAMES      period,      // Période
   ENUM_APPLIED_VOLUME  applied      // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_VOLUMES for CiVolumes).

© 2000-2015, MetaQuotes Software Corp.

Indicateurs de Bill Williams

Cet ensemble de chapitres contient les détails techniques des classes des indicateurs de Bill Williams de la Bibliothèque Standard MQL5 et les descriptions de ses composants principaux.

Classe/groupe	Description
CiAC	Oscillateur d'Accélération
CiAlligator	Alligator
CiAO	Oscillateur Awesome
CiFractals	Fractals
CiGator	Gator Oscillator
CiBWMFI	Market Facilitation Index

© 2000-2015, MetaQuotes Software Corp.

CiAC

CiAC est une classe permettant d'utiliser l'oscillateur Accelerator.

Description

La classe CiAC fournit la création, la configuration et l'accès aux données de l'oscillateur Accelerator.

Déclaration

```
class CiAC: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,      // symbole
   ENUM_TIMEFRAMES  period       // Période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_AC pour CiAC).

© 2000-2015, MetaQuotes Software Corp.

CiAlligator

CiAlligator est une classe permettant d'utiliser l'indicateur technique Alligator.

Description

La classe CiAlligator fournit la création, la configuration et l'accès aux données de l'indicateur Alligator.

Déclaration

```
class CiAlligator: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Méthodes de Classe

Attributs	
JawPeriod	Retourne la période de lissage pour la ligne de la Mâchoire
JawShift	Retourne le décalage horizontal de la ligne de la Mâchoire
TeethPeriod	Retourne la période de lissage de la ligne des Dents
TeethShift	Retourne le décalage horizontal de la ligne des Dents
LipsPeriod	Retourne la période de lissage de la ligne des Lévres
LipsShift	Retourne le décalage horizontal de la ligne des Lévres
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Jaw	Retourne l'élément du buffer de la ligne de la Mâchoire
Teeth	Retourne l'élément du buffer de la ligne des Dents
Lips	Retourne l'élément du buffer de la ligne des

© 2000-2015, MetaQuotes Software Corp.

	Lévres
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

JawPeriod

Retourne la période de lissage pour la ligne de la Mâchoire.

```
int  JawPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne de la Mâchoire, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

JawShift

Retourne le décalage horizontal de la ligne de la Mâchoire.

```
int  JawShift() const
```

Valeur de retour

Décalage horizontal de la ligne de la Mâchoire, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

TeethPeriod

Retourne la période de lissage de la ligne des Dents.

```
int  TeethPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne des Dents, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

TeethShift

Retourne le décalage horizontal de la ligne des Dents.

```
int  TeethShift() const
```

Valeur de retour

Décalage horizontal de la ligne des Dents, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

LipsPeriod

Retourne la période de lissage de la ligne des Lévres.

```
int  LipsPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne des Lévres, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

LipsShift

Retourne le décalage horizontal de la ligne des Lévres.

```
int  LipsShift() const
```

Valeur de retour

Décalage horizontal de la ligne des Lévres, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,          // symbole
   ENUM_TIMEFRAMES  period,          // Période
   int              jaw_period,      // Période de la Mâchoire
   int              jaw_shift,       // Décalage de la Mâchoire
   int              teeth_period,    // Période des Dents
   int              teeth_shift,     // Décalage des Dents
   int              lips_period,     // Période des Lévres
   int              lips_shift,      // Décalage des Lévres
   ENUM_MA_METHOD   ma_method,       // Méthode de lissage
   int              applied          // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

jaw_period

[in] Période de la Mâchoire.

jaw_shift

[in] Décalage de la Mâchoire.

teeth_period

[in] Période des Dents.

teeth_shift

[in] Décalage des Dents.

lips_period

[in] Période des Lévres.

lips_shift

[in] Décalage des Lévres.

ma_method

[in] Méthode de la moyenne mobile (énumération ENUM_MA_METHOD).

applied

[in] Type de volume à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Jaw

Retourne l'élément du buffer de la ligne de la Mâchoire de l'index spécifié.

```
double   Jaw(
   int   index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer de la ligne de la Mâchoire de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Teeth

Retourne l'élément du buffer de la ligne des Dents de l'index spécifié.

```
double  Teeth(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément buffer de la ligne des Dents de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Lips

Retourne l'élément du buffer de la ligne des Lévres de l'index spécifié.

```
double  Lips(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément buffer de la ligne des Lévres de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_ALLIGATOR for CiAlligator).

© 2000-2015, MetaQuotes Software Corp.

CiAO

CiAO est une classe permettant d'utiliser l'oscillateur Awesome.

Description

La classe CiAO fournit la création, la configuration et l'accès aux données de l'oscillateur Awesome.

Déclaration

```
class CiAO: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,     // symbole
   ENUM_TIMEFRAMES  period      // Période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_AO for CiAO).

© 2000-2015, MetaQuotes Software Corp.

CiFractals

La classe CiFractals permet d'utiliser l'indicateur technique Fractals.

Description

La classe CiFractals fournit la création, la configuration et l'accès aux données de l'indicateur Fractals.

Déclaration

```
class CiFractals: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Méthodes de Classe

Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Upper	Retourne l'élément du buffer supérieur
Lower	Retourne l'élément du buffer inférieur
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,     // symbole
   ENUM_TIMEFRAMES  period      // Période
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Upper

Retourne l'élément du buffer supérieur situé à l'index spécifié.

```
double  Upper(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer supérieur de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Lower

Retourne l'élément du buffer inférieur situé à l'index spécifié.

```
double  Lower(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer inférieur de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_FRACTALS for CiFractals).

© 2000-2015, MetaQuotes Software Corp.

CiGator

La classe CiFractals permet d'utiliser l'oscillateur Gator.

Description

La classe CiGator fournit la création, la configuration et l'accès aux données de l'oscillateur Gator.

Déclaration

```
class CiGator: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Méthodes de Classe

Attributs	
JawPeriod	Retourne la période de lissage pour la ligne de la Mâchoire
JawShift	Retourne le décalage horizontal de la ligne de la Mâchoire
TeethPeriod	Retourne la période de lissage de la ligne des Dents
TeethShift	Retourne le décalage horizontal de la ligne des Dents
LipsPeriod	Retourne la période de lissage de la ligne des Lévres
LipsShift	Retourne le décalage horizontal de la ligne des Lévres
MaMethod	Retourne la méthode de lissage
Applied	Retourne le type de prix ou le handle à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Upper	Retourne l'élément du buffer supérieur
Lower	Retourne l'élément du buffer inférieur
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

JawPeriod

Retourne la période de lissage pour la ligne de la Mâchoire.

```
int  JawPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne de la Mâchoire, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

JawShift

Retourne le décalage horizontal de la ligne de la Mâchoire.

```
int  JawShift() const
```

Valeur de retour

Décalage horizontal de la ligne de la Mâchoire, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

TeethPeriod

Retourne la période de lissage de la ligne des Dents.

```
int  TeethPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne des Dents, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

TeethShift

Retourne le décalage horizontal de la ligne des Dents.

```
int  TeethShift() const
```

Valeur de retour

Décalage horizontal de la ligne des Dents, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

LipsPeriod

Retourne la période de lissage de la ligne des Lévres.

```
int  LipsPeriod() const
```

Valeur de retour

Retourne la période de lissage pour la ligne des Lévres, définie à la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

LipsShift

Retourne le décalage horizontal de la ligne des Lévres.

```
int  LipsShift() const
```

Valeur de retour

Décalage horizontal de la ligne des Lévres, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

MaMethod

Retourne la méthode de lissage.

```
ENUM_MA_METHOD  MaMethod() const
```

Valeur de retour

Retourne la méthode de lissage, définie au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de prix ou le handle à appliquer.

```
int  Applied() const
```

Valeur de retour

Type de prix ou handle à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string           symbol,           // symbole
   ENUM_TIMEFRAMES  period,           // Période
   int              jaw_period,       // Période de la Mâchoire
   int              jaw_shift,        // Décalage de la Mâchoire
   int              teeth_period,     // Période des Dents
   int              teeth_shift,      // Décalage des Dents
   int              lips_period,      // Période des Lévres
   int              lips_shift,       // Décalage des Lévres
   ENUM_MA_METHOD   ma_method,        // Méthode de lissage
   int              applied           // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

jaw_period

[in] Période de la Mâchoire.

jaw_shift

[in] Décalage de la Mâchoire.

teeth_period

[in] Période des Dents.

teeth_shift

[in] Décalage des Dents.

lips_period

[in] Période des Lévres.

lips_shift

[in] Décalage des Lévres.

ma_method

[in] Méthode de lissage (énumération ENUM_MA_METHOD).

applied

[in] Type de volume à appliquer.

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Upper

Retourne l'élément du buffer supérieur situé à l'index spécifié.

```
double  Upper(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer supérieur de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Lower

Retourne l'élément du buffer inférieur situé à l'index spécifié.

```
double  Upper(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

L'élément du buffer inférieur de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_GATOR for CiGator).

© 2000-2015, MetaQuotes Software Corp.

CiBWMFI

CiBWMFI est une classe permettant d'utiliser l'indicateur technique Market Facilitation Index de Bill Williams.

Description

La classe CiBWMFI fournit la création, la configuration et l'accès aux données de l'indicateur Market Facilitation Index de Bill Williams.

Déclaration

```
class CiBWMFI: public CIndicator
```

Titre

```
#include <Indicators\BillWilliams.mqh>
```

Méthodes de Classe

Attributs	
Applied	Retourne le type de volume à appliquer
Méthodes de Création	
Create	Crée l'indicateur
Méthodes d'Accès aux Données	
Main	Retourne l'élément du buffer
Entrée/Sortie	
virtual Type	Retourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

Applied

Retourne le type de volume à appliquer.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valeur de retour

Type de volume à appliquer, défini au moment de la création de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée l'indicateur avec les paramètres spécifiés. Utiliser Refresh() et GetData() pour mettre à jour et récupérer les valeurs de l'indicateur.

```
bool  Create(
   string               symbol,      // symbole
   ENUM_TIMEFRAMES      period,      // Période
   ENUM_APPLIED_VOLUME  applied      // Type de volume à appliquer
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

applied

[in] Type de volume à appliquer (énumération ENUM_APPLIED_VOLUME).

Valeur de retour

vrai si réalisé avec succès, faux si l'indicateur n'a pas été créé.

© 2000-2015, MetaQuotes Software Corp.

Main

Retourne l'élément du buffer de l'index spécifié.

```
double  Main(
   int  index      // Index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

Element buffer de l'index spécifié en cas de succès, ou EMPTY_VALUE si aucune donnée correcte n'existe.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_BWMFI for CiBWMFI).

© 2000-2015, MetaQuotes Software Corp.

CiCustom

CiCustom est une classe prévue pour utiliser les indicateurs techniques personnalisés.

Description

La classe CiCustom fournit la création, la configuration et l'accès aux données de l'indicateur personnalisés.

Déclaration

```
class CiCustom: public CIndicator
```

Titre

```
#include <Indicators\Custom.mqh>
```

Méthodes de Classe

Attributs	
NumBuffers	Définit le nombre de buffers
NumParams	Retourne le nombre de paramètres
ParamType	Retourne le type du paramètre spécifié
ParamLong	Retourne la valeur du paramètre spécifié de type integer
ParamDouble	Retourne la valeur du paramètre spécifié de type double
ParamString	Retourne la valeur du paramètre spécifié de type string
Entrée/Sortie	
virtual Type	REtourne l'identifiant du type de l'objet

© 2000-2015, MetaQuotes Software Corp.

NumBuffers

Définit le nombre de buffers.

```
bool  NumBuffers(
   int  buffers      // nombre de buffers
   )
```

Valeur de retour

vrai si réalisé avec succès, faux si les buffers n'ont pas été définis.

© 2000-2015, MetaQuotes Software Corp.

NumParams

Retourne le nombre de paramètres

```
int  NumParams() const
```

Valeur de retour

Nombre de paramètres utilisés pour créer l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

ParamType

Retourne le type du paramètre à l'index spécifié.

```
ENUM_DATATYPE  ParamType(
   int  index      // index du paramètre
   ) const
```

Paramètres

index

[in] Index du paramètre.

Valeur de retour

Retourne le type de donnée (valeur de l'énumération ENUM_DATATYPE) du paramètre à l'index spécifié, utilisé pour créer l'indicateur.

Note

Si l'index du paramètre est invalide, retourne WRONG_VALUE.

© 2000-2015, MetaQuotes Software Corp.

ParamLong

Retourne la valeur du paramètre spécifié de type long.

```
long  ParamLong(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index du paramètre.

Valeur de retour

La valeur du paramètre spécifié de type long utilisé pour créer l'indicateur.

Note

Si le nombre est invalide, retourne 0.

© 2000-2015, MetaQuotes Software Corp.

ParamDouble

Retourne la valeur du paramètre spécifié de type double.

```
double  ParamDouble(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index du paramètre.

Valeur de retour

La valeur du paramètre spécifié de type double utilisé pour créer l'indicateur.

Note

Si le nombre est invalide, retourne EMPTY_VALUE.

© 2000-2015, MetaQuotes Software Corp.

ParamString

Retourne la valeur du paramètre spécifié de type string.

```
string  ParamString(
   int  index      // index
   ) const
```

Paramètres

index

[in] Index du paramètre.

Valeur de retour

La valeur du paramètre spécifié de type string utilisé pour créer l'indicateur.

Note

Si le nombre est invalide, retourne une chaîne vide.

© 2000-2015, MetaQuotes Software Corp.

Type

Retourne l'identifiant du type de l'objet.

```
virtual int  Type() const
```

Valeur de retour

Identifiant du type de l'objet (IND_CUSTOM for CiCustom).

© 2000-2015, MetaQuotes Software Corp.

Classes de Trade

Cette section contient les détails techniques d'utilisation des classes de trade et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création des programmes personnalisés (Expert Advisors).

La Bibliothèque Standard MQL5 (en terme d'ensemble de données) est située dans le répertoire de travail du terminal dans le répertoire Include\Arrays.

Classe/Groupe	Description
CAccountInfo	Classe pour travailler avec les propriétés du compte de trading
CSymbolInfo	Classe pour travailler avec les propriétés des instruments de trading
COrderInfo	Classe pour travailler avec les propriétés des ordres en attente
CHistoryOrderInfo	Classe pour travailler avec les propriétés de l'historique des ordres
CPositionInfo	Classe pour travailler avec les propriétés des positions ouvertes
CDealInfo	Classe pour travailler avec les propriétés de l'historique des deals
CTrade	Classe pour l'exécution des opérations de trading
CTerminalInfo	Classe pour récupérer les propriétés de l'environnement du programme MQL5

© 2000-2015, MetaQuotes Software Corp.

CAccountInfo

La classe CAccountInfo permet d'accéder aux propriétés du compte des positions actuellement ouvertes.

Description

La classe CAccountInfo permet d'accéder aux propriétés du compte des positions actuellement ouvertes.

Déclaration

```
class CAccountInfo : public CObject
```

Titre

```
#include <Trade\AccountInfo.mqh>
```

Méthodes de classe par groupes

Accès aux propriétés de type integer	
Login	Retourne le numéro du compte
TradeMode	Retourne le mode de trading
TradeModeDescription	Retourne le mode de trading sous forme d'une chaîne de caractères
Leverage	Retourne le levier actuel
MarginMode	Retourne le mode de stop out du compte
MarginModeDescription	Retourne la description du mode de stop out du compte
TradeAllowed	Retourne le flag de trade allowance
TradeExpert	Retourne le flag de trade allowance automatique
LimitOrders	Retourne le nombre maximal d'ordres en attente autorisés
Accès aux propriétés de type double	
Balance	Retourne la balance du compte
Credit	Retourne le montant du crédit
Profit	Retourne le montant du profit actuel du compte
Equity	Retourne le montant actuel du compte
Margin	Retourne le montant de la marge réservée
FreeMargin	Retourne le montant de la marge libre

© 2000-2015, MetaQuotes Software Corp.

MarginLevel	Retourne le montant de la marge
MarginCall	Retourne le montant de la marge de dépôt
MarginStopOut	Retourne le niveau de marge de Stop Out
Accès aux propriétés de type texte	
Name	Retourne le nom du client
Server	Retourne le nom du serveur de trading
Currency	Retourne le nom de devise actuelle
Company	Retourne le nom de la compagnie qui fournit le compte
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer
InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string
Méthodes supplémentaires	
OrderProfitCheck	Retourne le profit évalué, basé sur les paramètres passés
MarginCheck	Retourne le montant de la marge requise pour exécuter une opération de trading
FreeMarginCheck	Retourne le montant de la marge libre restante après exécution d'une opération de trading
MaxLotCheck	Retourne le volume maximal possible d'une opération de trading

© 2000-2015, MetaQuotes Software Corp.

Login

Retourne le numéro du compte.

```
long  Login() const
```

Valeur de retour

Numéro du compte.

© 2000-2015, MetaQuotes Software Corp.

TradeMode

Retourne le mode de trading.

```
ENUM_ACCOUNT_TRADE_MODE  TradeMode() const
```

Valeur de retour

Mode de trading (valeur de l'énumération ENUM_ACCOUNT_TRADE_MODE).

© 2000-2015, MetaQuotes Software Corp.

TradeModeDescription

Retourne le mode de trading sous forme d'une chaîne de caractères.

```
string  TradeModeDescription() const
```

Valeur de retour

Mode de trading sous forme d'une chaîne de caractères.

© 2000-2015, MetaQuotes Software Corp.

Leverage

Retourne le niveau du levier actuel.

```
long  Leverage() const
```

Valeur de retour

Niveau de levier actuel.

© 2000-2015, MetaQuotes Software Corp.

MarginMode

Retourne le mode de Stop Out du compte (stop automatique en cas de perte totale du montant du compte).

```
ENUM_ACCOUNT_STOPOUT_MODE  MarginMode() const
```

Valeur de retour

Mode de Stop Out du compte (valeur de l'énumération ENUM_ACCOUNT_STOPOUT_MODE).

© 2000-2015, MetaQuotes Software Corp.

MarginModeDescription

Retourne le mode de définition du niveau de marge minimum sous forme d'une chaîne de caractères.

```
string  MarginModeDescription() const
```

Valeur de retour

Le mode de définition du niveau de marge minimum sous forme d'une chaîne de caractère.

© 2000-2015, MetaQuotes Software Corp.

TradeAllowed

Retourne le flag de trade allowance

```
bool  TradeAllowed() const
```

Valeur de retour

Flag de trade allowance

© 2000-2015, MetaQuotes Software Corp.

TradeExpert

Retourne le flag de trade allowance automatique.

```
bool  TradeExpert() const
```

Valeur de retour

Flag de trade allowance automatique.

© 2000-2015, MetaQuotes Software Corp.

LimitOrders

Retourne le nombre maximal d'ordres en attente autorisés

```
int  LimitOrders() const
```

Valeur de retour

Le nombre maximal d'ordres en attente autorisés.

Note

0 - aucune limite.

© 2000-2015, MetaQuotes Software Corp.

Balance

Retourne la balance du compte.

```
double  Balance() const
```

Valeur de retour

La balance du compte (dans la devise du compte).

© 2000-2015, MetaQuotes Software Corp.

Credit

Retourne le montant du crédit.

```
double  Credit() const
```

Valeur de retour

Montant du crédit (dans la devise du compte).

© 2000-2015, MetaQuotes Software Corp.

Profit

Retourne le montant du profit actuel du compte.

```
double  Profit() const
```

Valeur de retour

Montant actuel du profit compte (dans la devise du compte).

© 2000-2015, MetaQuotes Software Corp.

Equity

Retourne le montant actuel du compte.

```
double  Equity() const
```

Valeur de retour

Montant actuel du compte (dans la devise du compte).

© 2000-2015, MetaQuotes Software Corp.

Margin

Retourne le montant de la marge réservée.

```
double  Margin() const
```

Valeur de retour

Montant de la marge réservée (dans la devise du compte).

© 2000-2015, MetaQuotes Software Corp.

FreeMargin

Retourne le montant de la marge libre.

```
double  FreeMargin() const
```

Valeur de retour

Montant de la marge libre (dans la devise du compte).

© 2000-2015, MetaQuotes Software Corp.

MarginLevel

Retourne le niveau de marge.

```
double  MarginLevel() const
```

Valeur de retour

Niveau de marge.

© 2000-2015, MetaQuotes Software Corp.

MarginCall

Retourne le montant de la marge de dépôt.

```
double  MarginCall() const
```

Valeur de retour

Montant de la marge de dépôt.

© 2000-2015, MetaQuotes Software Corp.

MarginStopOut

Retourne le niveau de marge de Stop Out.

```
double  MarginStopOut() const
```

Valeur de retour

Le niveau de marge de Stop Out.

© 2000-2015, MetaQuotes Software Corp.

Name

Retourne le nom du client.

```
string  Name() const
```

Valeur de retour

Nom du client.

© 2000-2015, MetaQuotes Software Corp.

Server

Retourne le nom du serveur de trading.

```
string  Server() const
```

Valeur de retour

Nom du serveur de trading.

© 2000-2015, MetaQuotes Software Corp.

Currency

Retourne le nom de devise actuelle.

```
string  Currency() const
```

Valeur de retour

Nom de la devise du compte.

© 2000-2015, MetaQuotes Software Corp.

Company

Retourne le nom de la société qui fournit le compte.

```
string  Company() const
```

Valeur de retour

Nom de la société fournissant le compte.

© 2000-2015, MetaQuotes Software Corp.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
long  InfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  prop_id     // Identifiant de la propriété
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété. La valeur peut être une des valeurs de l'énumération ENUM_ACCOUNT_INFO_INTEGER .

Valeur de retour

Valeur de type long.

© 2000-2015, MetaQuotes Software Corp.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
double  InfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  prop_id     // Identifiant de la propriété
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété. La valeur peut être une des valeurs de l'énumération ENUM_ACCOUNT_INFO_DOUBLE .

Valeur de retour

Valeur de type double.

© 2000-2015, MetaQuotes Software Corp.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
string  InfoString(
   ENUM_ACCOUNT_INFO_STRING  prop_id     // Identifiant de la propriété
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété. La valeur peut être une des valeurs de l'énumération ENUM_ACCOUNT_INFO_STRING .

Valeur de retour

Valeur de type string.

© 2000-2015, MetaQuotes Software Corp.

OrderProfitCheck

La fonction calcule le profit du compte courant basé sur les paramètres passés. La fonction est utilisée pour pré-évaluer le résultat d'une opération de trading. La valeur est retournée dans la devise du compte.

```
double  OrderProfitCheck(
   const string        symbol,              // symbole
   ENUM_ORDER_TYPE     trade_operation,     // opération type (ORDER_TYPE_BUY or ORDER
   double              volume,              // volume
   double              price_open,          // prix d'ouverture
   double              price_close          // prix de clôture
   ) const
```

Paramètres

symbol

[in] Symbole de l'opération de trading.

trade_operation

[in] Type de l'opération de trading (énumération ***ENUM_ORDER_TYPEEMPTY_VALUE en cas d'erreur.

© 2000-2015, MetaQuotes Software Corp.

MarginCheck

Retourne le montant de la marge requise pour exécuter une opération de trading.

```
double  MarginCheck(
   const string        symbol,              // symbole
   ENUM_ORDER_TYPE     trade_operation,     // opération
   double              volume,              // volume
   double              price                // prix
   ) const
```

Paramètres

symbol

[in] Symbole de l'opération de trading.

trade_operation

[in] Type de l'opération de trading (énumération ***ENUM_ORDER_TYPE

© 2000-2015, MetaQuotes Software Corp.

FreeMarginCheck

Retourne le montant de la marge libre restante après une opération de trading.

```
double  FreeMarginCheck(
   const string         symbol,              // symbole
   ENUM_ORDER_TYPE      trade_operation,     // opération
   double               volume,              // volume
   double               price                // prix
   ) const
```

Paramètres

symbol

[in] Symbole de l'opération de trading.

trade_operation

[in] Type de l'opération de trading (énumération ***ENUM_ORDER_TYPE

© 2000-2015, MetaQuotes Software Corp.

MaxLotCheck

Gets the maximum possible Volume de l'opération de trading.

```
double  MaxLotCheck(
   const string        symbol,              // symbole
   ENUM_ORDER_TYPE     trade_operation,     // opération
   double              price,               // prix
   double              percent=100          // pourcentage de la marge disponible (0-1
   ) const
```

Paramètres

symbol

[in] Symbole de l'opération de trading.

trade_operation

[in] Type de l'opération de trading (énumération ***ENUM_ORDER_TYPE

© 2000-2015, MetaQuotes Software Corp.

CSymbolInfo

La classe CSymbolInfo permet d'accéder aux propriétés du symbole.

Description

La classe CSymbolInfo fournit l'accès aux propriétés du symbole.

Déclaration

```
class CSymbolInfo : public CObject
```

Titre

```
#include <Trade\SymbolInfo.mqh>
```

Méthodes de classe par groupes

Contrôle	
Refresh	Rafraîchit les données du symbole
RefreshRates	Rafraîchit les cotations du symbole
Propriétés	
Name	Retourne/Définit le nom du symbole
Select	Retourne/Définit le flag "Market Watch" du symbole
IsSynchronized	Vérifie la synchronisation du symbole avec le serveur
Volumes	
Volume	Retourne le volume du dernier deal
VolumeHigh	Retourne le volume maximal d'une journée
VolumeLow	Retourne le volume minimal d'une journée
Divers	
Time	Retourne l'heure de la dernière cotation
Spread	Retourne le montant du spread (en points)
SpreadFloat	Retourne le flag de spread flottant
TicksBookDepth	Retourne la profondeur des ticks
Niveaux	
StopsLevel	Retourne l'incrément minimal pour les ordres (en points)
FreezeLevel	Retourne la distance de gel des opérations de trading (en points)

© 2000-2015, MetaQuotes Software Corp.

Prix de l'Offre (Bid)	
Bid	Retourne le prix de l'Offre (Bid) courant
BidHigh	Retourne le prix Bid maximal d'une journée
BidLow	Retourne le prix Bid minimal d'une journée
Prix de la Demande (Ask)	
Ask	Retourne le prix de la Demande (Ask) courant
AskHigh	Retourne le prix Ask maximal d'une journée
AskLow	Retourne le prix Ask minimal d'une journée
Prix	
Last	Retourne le prix Last courant
LastHigh	Retourne le prix Last maximal d'une journée
LastLow	Retourne le prix Last minimal d'une journée
Modes de trading	
TradeCalcMode	Retourne le mode de calcul du coût d'un contrat
TradeCalcModeDescription	Retourne le mode de calcul du coût d'un contrat sous forme d'une chaîne de caractères
TradeMode	Retourne le type d'exécution d'un ordre
TradeModeDescription	Retourne le type d'exécution de l'ordre sous forme d'une chaîne de caractères
TradeExecution	Retourne le mode de clôture des deals
TradeExecutionDescription	Retourne le mode de clôture des deals sous forme d'une chaîne de caractères
Swaps	
SwapMode	Retourne le modèle de calcul du swap
SwapModeDescription	Retourne le modèle de calcul du swap sous forme d'une chaîne de caractères
SwapRollover3days	Retourne le coût de 3 jours de swap
SwapRollover3daysDescription	Retourne le coût de 3 jours de swap sous forme d'une chaîne de caractères
Marge et flags	
MarginInitial	Retourne la valeur de la marge initiale
MarginMaintenance	Retourne la valeur de la marge de maintenance
MarginLong	Retourne le taux de marge appliqué pour les positions longues
MarginShort	Retourne le taux de marge appliqué pour les

© 2000-2015, MetaQuotes Software Corp.

	positions courtes
MarginLimit	Retourne le taux de marge appliqué pour les ordres Limit
MarginStop	Retourne le taux de marge appliqué pour les ordres Stop
MarginStopLimit	Retourne le taux de marge appliqué pour les ordres StopLimit
TradeTimeFlags	Retourne les flags des modes autorisés d'expiration d'un ordre
TradeFillFlags	Retourne les flags des modes autorisés de remplissage d'un ordre
Quantification	
Digits	Retourne le nombre de décimales (nombre de chiffres après la virgule)
Point	Retourne la valeur d'un point
TickValue	Retourne le coût d'un tick (changement minimal du prix)
TickValueProfit	Retourne le prix calculé du tick pour une position gagnante
TickValueLoss	Retourne le prix calculé du tick pour une position perdante
TickSize	Retourne le changement minimal du prix
Tailles des contrats	
ContractSize	Retourne le montant du contrat de trading
LotsMin	Retourne le volume minimal pour clôturer un deal
LotsMax	Retourne le volume maximal pour clôturer un deal
LotsStep	Retourne le pas minimal de changement de volume pour clôturer un deal
LotsLimit	Retourne le volume maximal autorisé pour une position ouverte et pour les ordres en attente (sans distinction de la direction) pour un symbole
Tailles de swaps	
SwapLong	Retourne la valeur de swap d'une position longue
SwapShort	Retourne la valeur de swap d'une position courte

© 2000-2015, MetaQuotes Software Corp.

Propriétés texte	
CurrencyBase	Retourne le nom de la devise de base du symbole
CurrencyProfit	Retourne le nom de la devise de profit
CurrencyMargin	Retourne le nom de la devise de marge
Bank	Retourne le nom de la source de la cotation actuelle
Description	Retourne la description du symbole sous forme d'une chaîne de caractères
Path	Retourne le chemin dans l'arbre des symboles
Propriétés du symbole	
SessionDeals	Retourne le nombre de deals de la session courante
SessionBuyOrders	Retourne le nombre d'ordres Buy (ordres d'achat) au moment de l'appel
SessionSellOrders	Retourne le nombre d'ordres Sell (ordres de vente) au moment de l'appel
SessionTurnover	Retourne le résumé du turnover de la session courante
SessionInterest	Retourne le résumé de l'open interest de la session courante
SessionBuyOrdersVolume	Retourne le volume courant des ordres Buy (ordres d'achat)
SessionSellOrdersVolume	Retourne le volume courant des ordres Sell (ordres de vente)
SessionOpen	Retourne le prix d'ouverture de la session courante
SessionClose	Retourne le prix de clôture de la session courante
SessionAW	Retourne le prix moyen pondéré de la session courante
SessionPriceSettlement	Retourne le prix de compensation (settlement price) de la session courante
SessionPriceLimitMin	Retourne le prix minimal de la session courante
SessionPriceLimitMax	Retourne le prix maximal de la session courante
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer

© 2000-2015, MetaQuotes Software Corp.

InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string.
Fonctions d'aide	
NormalizePrice	Retourne la valeur du prix, normalisée en utilisant les propriétés du symbole

© 2000-2015, MetaQuotes Software Corp.

Refresh

Rafraîchit les données du symbole.

```
void  Refresh()
```

Valeur de retour

Aucune.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

RefreshRates

Rafraîchit les cotations du symbole.

```
bool  RefreshRates()
```

Valeur de retour

vrai en cas de succès, faux si les cotations ne peuvent pas être rafraîchies.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Name

Retourne le nom du symbole.

```
string  Name() const
```

Valeur de retour

nom du symbole.

Name

Définit le nom du symbole.

```
bool  Name(string name)
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Select

Retourne le flag "Market Watch" du symbole.

```
bool  Select() const
```

Valeur de retour

Le flag "Market Watch" du symbole.

Select

Définit le flag "Market Watch" du symbole.

```
bool  Select()
```

Valeur de retour

vrai en cas de succès, faux si le flag n'a pas pu être modifié.

© 2000-2015, MetaQuotes Software Corp.

IsSynchronized

Vérifie la synchronisation du symbole avec le serveur.

```
bool  IsSynchronized() const
```

Valeur de retour

vrai si le symbole est synchronisé avec le serveur, sinon faux.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Volume

Retourne le volume du dernier deal.

```
long  Volume() const
```

Valeur de retour

Le volume du dernier deal.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

VolumeHigh

Retourne le volume maximal d'une journée.

```
long  VolumeHigh() const
```

Valeur de retour

Le volume maximal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

VolumeLow

Retourne le volume minimal d'une journée.

```
long  VolumeLow() const
```

Valeur de retour

Le volume minimal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Time

Retourne l'heure de la dernière cotation.

```
datetime  Time() const
```

Valeur de retour

L'heure de la dernière cotation.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Spread

Retourne le montant du spread (en points).

```
int  Spread() const
```

Valeur de retour

Le montant du spread (en points).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SpreadFloat

Retourne le flag de spread flottant.

```
bool  SpreadFloat() const
```

Valeur de retour

Le flag de spread flottant.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TicksBookDepth

Retourne la profondeur de sauvegarde des ticks.

```
int  TicksBookDepth() const
```

Valeur de retour

La profondeur de sauvegarde des ticks.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

StopsLevel

Retourne le niveau minimal du stop pour les ordres (en points).

```
int  StopsLevel() const
```

Valeur de retour

Le niveau minimal du stop pour les ordres (en points).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

FreezeLevel

Retourne le montant de gel (en points)

```
int  FreezeLevel() const
```

Valeur de retour

Distance du niveau de gel (en points).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Bid

Retourne le prix de l'Offre (Bid) courant.

```
double  Bid() const
```

Valeur de retour

Le prix de l'Offre (Bid) courant.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

BidHigh

Retourne le prix Bid maximal d'une journée.

```
double  BidHigh() const
```

Valeur de retour

Le prix Bid maximal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

BidLow

Retourne le prix Bid minimal d'une journée.

```
double  BidLow() const
```

Valeur de retour

Le prix Bid minimal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Ask

Retourne le prix de la Demande (Ask) courant.

```
double  Ask() const
```

Valeur de retour

Prix de la Demande (Ask)

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

AskHigh

Retourne le prix Ask maximal d'une journée.

```
double  AskHigh() const
```

Valeur de retour

Le prix Ask maximal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

AskLow

Retourne le prix Ask minimal d'une journée.

```
double  AskLow() const
```

Valeur de retour

Le prix Ask minimal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Last

Retourne le prix Last courant.

```
double  Last() const
```

Valeur de retour

Prix Last courant.

© 2000-2015, MetaQuotes Software Corp.

LastHigh

Retourne le prix Last maximal d'une journée.

```
double  LastHigh() const
```

Valeur de retour

Le prix Last maximal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

LastLow

Retourne le prix Last minimal d'une journée.

```
double  LastLow() const
```

Valeur de retour

Le prix Last minimal d'une journée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TradeCalcMode

Retourne le mode de calcul du coût d'un contrat.

```
ENUM_SYMBOL_CALC_MODE  TradeCalcMode() const
```

Valeur de retour

Le mode de calcul du coût d'un contrat (valeur de l'énumération ENUM_SYMBOL_CALC_MODE).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TradeCalcModeDescription

Retourne le mode de calcul du coût d'un contrat sous forme d'une chaîne de caractères.

```
string  TradeCalcModeDescription() const
```

Valeur de retour

Le mode de calcul du coût d'un contrat sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TradeMode

Retourne le type d'exécution d'un ordre.

```
ENUM_SYMBOL_TRADE_MODE  TradeMode() const
```

Valeur de retour

Le type d'exécution d'un ordre (valeur de l'énumération ENUM_SYMBOL_TRADE_MODE).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TradeModeDescription

Retourne le mode de trading sous forme d'une chaîne de caractères.

```
string  TradeModeDescription() const
```

Valeur de retour

Mode de trading sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TradeExecution

Retourne le mode d'exécution d'un trade.

```
ENUM_SYMBOL_TRADE_EXECUTION  TradeExecution() const
```

Valeur de retour

Le mode d'exécution d'un trade (valeur de l'énumération ENUM_SYMBOL_TRADE_EXECUTION).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TradeExecutionDescription

Retourne le mode d'exécution d'un trade sous forme d'une chaîne de caractères.

```
string  TradeExecutionDescription() const
```

Valeur de retour

Le mode d'exécution d'un trade sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SwapMode

Retourne le modèle de calcul du swap.

```
ENUM_SYMBOL_SWAP_MODE  SwapMode() const
```

Valeur de retour

Le modèle de calcul du swap (valeur de l'énumération ENUM_SYMBOL_SWAP_MODE).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SwapModeDescription

Retourne le modèle de calcul du swap sous forme d'une chaîne de caractères.

```
string  SwapModeDescription() const
```

Valeur de retour

Le modèle de calcul du swap sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SwapRollover3days

Retourne le jour du retournement du swap.

```
ENUM_DAY_OF_WEEK  SwapRollover3days() const
```

Valeur de retour

Le jour de retournement du swap (valeur de l'énumération ENUM_DAY_OF_WEEK).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SwapRollover3daysDescription

Retourne le jour du retournement du swap sous forme d'une chaîne de caractères.

```
string  SwapRollover3daysDescription() const
```

Valeur de retour

Le jour du retournement du swap sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

MarginInitial

Retourne la valeur de la marge initiale.

```
double  MarginInitial()
```

Valeur de retour

La valeur de la marge initiale.

Note

Correspond au montant de la marge (dans la devise de marge de l'instrument) qui est chargé pour un lot. Utilisé pour vérifier le capital du client lorsqu'il entre sur le marché.

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

MarginMaintenance

Retourne la valeur de la marge de maintenance.

```
double  MarginMaintenance()
```

Valeur de retour

La valeur de la marge de maintenance.

Note

Correspond au montant de la marge (dans la devise de marge de l'instrument) qui est chargé pour un lot. Utilisé pour vérifier le capital du client lorsque l'état du compte est modifié. Si la marge de maintenance est égale à 0, la marge initiale est alors utilisée.

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

MarginLong

Retourne le taux de marge appliqué pour les positions longues.

```
double  MarginLong() const
```

Valeur de retour

Le taux de marge appliqué pour les positions longues.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

MarginShort

Retourne le taux de marge appliqué pour les positions courtes.

```
double  MarginShort() const
```

Valeur de retour

Le taux de marge appliqué pour les positions courtes.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

MarginLimit

Retourne le taux de marge appliqué aux ordres Limit.

```
double  MarginLimit() const
```

Valeur de retour

Le taux de marge appliqué aux ordres Limit.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

MarginStop

Retourne le taux de marge appliqué aux ordres Stop.

```
double  MarginStop() const
```

Valeur de retour

Le taux de marge appliqué aux ordres Stop.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

MarginStopLimit

Retourne le taux de marge appliqué aux ordres Stop Limit.

```
double  MarginStopLimit() const
```

Valeur de retour

Le taux de marge appliqué aux ordres Stop Limit.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TradeTimeFlags

Retourne les flags des modes autorisés d'expiration d'un ordre.

```
int  TradeTimeFlags() const
```

Valeur de retour

Les flags des modes autorisés d'expiration d'un ordre.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TradeFillFlags

Retourne les flags des modes autorisés de remplissage d'un ordre.

```
int  TradeFillFlags() const
```

Valeur de retour

Les flags des modes autorisés de remplissage d'un ordre.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Digits

Retourne le nombre de décimales (nombre de chiffres après la virgule).

```
int  Digits() const
```

Valeur de retour

Retourne le nombre de décimales (nombre de chiffres après la virgule).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Point

Retourne la valeur d'un point.

```
double  Point() const
```

Valeur de retour

Valeur d'un point.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TickValue

Retourne le coût d'un tick (changement minimal du prix).

```
double  TickValue() const
```

Valeur de retour

Le coût d'un tick (changement minimal du prix).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TickValueProfit

Retourne le prix calculé du tick pour une position gagnante.

```
double  TickValueProfit() const
```

Valeur de retour

Le prix calculé du tick pour une position gagnante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TickValueLoss

Retourne le prix calculé du tick pour une position perdante.

```
double  TickValueLoss() const
```

Valeur de retour

Le prix calculé du tick pour une position perdante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

TickSize

Retourne le changement minimal du prix.

```
double  TickSize() const
```

Valeur de retour

Le changement minimal du prix.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

ContractSize

Retourne le montant du contrat de trading.

```
double  ContractSize() const
```

Valeur de retour

Le montant du contrat de trading.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

LotsMin

Retourne le volume minimal pour clôturer un deal.

```
double  LotsMin() const
```

Valeur de retour

Le volume minimal pour clôturer un deal.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

LotsMax

Retourne le volume maximal pour clôturer un deal.

```
double  LotsMax() const
```

Valeur de retour

Le volume maximal pour clôturer un deal.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

LotsStep

Retourne le pas minimal de changement de volume pour clôturer un deal.

```
double  LotsStep() const
```

Valeur de retour

Le pas minimal de changement de volume pour clôturer un deal.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

LotsLimit

Retourne le volume maximal autorisé pour une position ouverte et pour les ordres en attente (sans distinction de la direction) pour un symbole.

```
double  LotsLimit() const
```

Valeur de retour

Le volume maximal autorisé pour une position ouverte et pour les ordres en attente (sans distinction de la direction) pour un symbole.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SwapLong

Retourne la valeur de swap d'une position longue.

```
double  SwapLong() const
```

Valeur de retour

La valeur de swap d'une position longue.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SwapShort

Retourne la valeur de swap d'une position courte.

```
double  SwapShort() const
```

Valeur de retour

La valeur de swap d'une position courte.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

CurrencyBase

Retourne le nom de la devise de base du symbole.

```
string  CurrencyBase() const
```

Valeur de retour

Le nom de la devise de base du symbole.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

CurrencyProfit

Retourne le nom de la devise de profit.

```
string  CurrencyProfit() const
```

Valeur de retour

Le nom de la devise de profit.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

CurrencyMargin

Retourne le nom de la devise de marge.

```
string  CurrencyMargin() const
```

Valeur de retour

Le nom de la devise de marge.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Bank

Retourne le nom de la source de la cotation actuelle.

```
string  Bank() const
```

Valeur de retour

Le nom de la source de la cotation actuelle.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Description

Retourne la description du symbole sous forme d'une chaîne de caractères.

```
string  Description() const
```

Valeur de retour

La description du symbole sous forme d'une chaîne de caractères.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

Path

Le chemin dans l'arbre des symboles.

```
string  Path() const
```

Valeur de retour

Le chemin dans l'arbre des symboles.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionDeals

Retourne le nombre de deals de la session courante.

```
long  SessionDeals() const
```

Valeur de retour

Retourne le nombre de deals de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionBuyOrders

Retourne le nombre d'ordres Buy (ordres d'achat) au moment de l'appel.

```
long  SessionBuyOrders() const
```

Valeur de retour

Le nombre d'ordres Buy (ordres d'achat) au moment de l'appel.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionSellOrders

Retourne le nombre d'ordres Sell (ordres de vente) au moment de l'appel.

```
long  SessionSellOrders() const
```

Valeur de retour

Le nombre d'ordres Sell (ordres de vente) au moment de l'appel.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionTurnover

Retourne le résumé du turnover de la session courante.

```
double  SessionTurnover() const
```

Valeur de retour

Le résumé du turnover de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionInterest

Retourne le résumé de l'open interest de la session courante.

```
double  SessionInterest() const
```

Valeur de retour

Le résumé de l'open interest de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionBuyOrdersVolume

Retourne le volume courant des ordres Buy (ordres d'achat).

```
double  SessionBuyOrdersVolume() const
```

Valeur de retour

Le volume courant des ordres Buy (ordres d'achat).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionSellOrdersVolume

Retourne le volume courant des ordres Sell (ordres de vente).

```
double  SessionSellOrdersVolume() const
```

Valeur de retour

Le volume courant des ordres Sell (ordres de vente).

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionOpen

Retourne le prix d'ouverture de la session courante.

```
double  SessionOpen() const
```

Valeur de retour

Le prix d'ouverture de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionClose

Retourne le prix de clôture de la session courante.

```
double  SessionClose() const
```

Valeur de retour

Le prix de clôture de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionAW

Retourne le prix moyen pondéré de la session courante.

```
double  SessionAW() const
```

Valeur de retour

Le prix moyen pondéré de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionPriceSettlement

Retourne le prix de compensation (settlement price) de la session courante.

```
double  SessionPriceSettlement() const
```

Valeur de retour

Le prix de compensation (settlement price) de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionPriceLimitMin

Retourne le prix minimal de la session courante.

```
double  SessionPriceLimitMin() const
```

Valeur de retour

Le prix minimal de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

SessionPriceLimitMax

Retourne le prix maximal de la session courante.

```
double  SessionPriceLimitMax() const
```

Valeur de retour

Le prix maximal de la session courante.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
bool  InfoInteger(
   ENUM_SYMBOL_INFO_INTEGER  prop_id,     // Identifiant de la propriété
   long&                     var          // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer (valeur de l'énumération ENUM_SYMBOL_INFO_INTEGER<t3>).</t3>

var

[out] Référence à une variable de type long pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
bool  InfoDouble(
   ENUM_SYMBOL_INFO_DOUBLE  prop_id,     // Identifiant de la propriété
   double&                  var          // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double (valeur de l'énumération ENUM_SYMBOL_INFO_DOUBLE<t3>).</t3>

var

[out] Référence à une variable de type double pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
bool  InfoString(
   ENUM_SYMBOL_INFO_STRING  prop_id,     // Identifiant de la propriété
   string&                  var          // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété texte.

var

[out] Référence à une variable de type string pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

NormalizePrice

Retourne la valeur du prix, normalisée en utilisant les propriétés du symbole.

```
double  NormalizePrice(
   double  price   // prix
   ) const
```

Paramètres

price

[in] Prix.

Valeur de retour

Prix normalisé.

Note

Le symbole doit être sélectionné avec la méthode Name.

© 2000-2015, MetaQuotes Software Corp.

COrderInfo

COrderInfo est la classe d'accès aux propriétés des ordres en attente.

Description

La classe COrderInfo permet d'accéder aux propriétés d'un ordre en attente.

Déclaration

```
class COrderInfo : public CObject
```

Titre

```
#include <Trade\OrderInfo.mqh>
```

Méthodes de classe par groupes

Accès aux propriétés de type integer	
Ticket	Retourne le ticket d'un ordre sélectionné auparavant
TimeSetup	Retourne l'heure de placement d'un ordre
TimeSetupMsc	Retourne l'heure de placement d'un ordre en millisecondes depuis le 01/01/1970
OrderType	Retourne le type de l'ordre
OrderTypeDescription	Retourne le type de l'ordre sous forme d'une chaîne de caractères
Etat	Retourne l'état de l'ordre
StateDescription	Retourne l'état de l'ordre sous forme d'une chaîne de caractères
TimeExpiration	Retourne l'heure d'expiration d'un ordre
TimeDone	Retourne l'heure d'exécution ou d'annulation d'un ordre
TimeDoneMsc	Retourne l'heure d'exécution ou d'annulation d'un ordre en millisecondes depuis le 01/01/1970
TypeFilling	Retourne le type d'exécution de l'ordre par ordre
TypeFillingDescription	Retourne le type d'exécution de l'ordre par ordre sous forme d'une chaîne de caractères
TypeTime	Retourne le type de l'ordre à l'heure d'expiration
TypeTimeDescription	Retourne le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères

© 2000-2015, MetaQuotes Software Corp.

Magic	Retourne l'identifiant de l'expert ayant placé l'ordre
PositionId	Retourne l'identifiant de la position
Accès aux propriétés de type double	
VolumeInitial	Retourne le volume initial de l'ordre
VolumeCurrent	Retourne le volume non atteint de l'ordre
PriceOpen	Retourne le prix de l'ordre
StopLoss	Retourne le Stop Loss de l'ordre
TakeProfit	Retourne le Take Profit de l'ordre
PriceCurrent	Retourne le prix courant du symbole
PriceStopLimit	Retourne le prix de l'ordre limite
Accès aux propriétés de type texte	
Symbol	Retourne le nom du symbole de l'ordre.
Comment	Retourne le commentaire de l'ordre
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer
InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string
Etat	
StoreState	Sauvegarde les paramètres de l'ordre
CheckState	Vérifie les paramères actuels par rapport aux paramètres sauvegardés
Sélection	
Select	Sélectionne un ordre par son ticket pour accéder par la suite à ses propriétés.
SelectByIndex	Sélectionne un ordre par son index pour accéder par la suite à ses propriétés.

© 2000-2015, MetaQuotes Software Corp.

Ticket

Retourne le ticket d'un ordre sélectionné précédemment en utilisant la méthode Select.

```
ulong  Ticket() const
```

Valeur de retour

Ticket de l'ordre en cas de succès, sinon ULONG_MAX.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeSetup

Retourne l'heure de placement de l'ordre.

```
datetime  TimeSetup() const
```

Valeur de retour

L'heure de placement de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeSetupMsc

Retourne l'heure de placement d'un ordre en millisecondes depuis le 01/01/1970<t2>.</t2>

```
ulong  TimeSetupMsc() const
```

Valeur de retour

L'heure de placement d'un ordre en millisecondes depuis le 01/01/1970.

Note

L'ordre doit tout d'abord être sélectionné pour pouvoir y accéder en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

OrderType

Retourne le type de l'ordre.

```
ENUM_ORDER_TYPE  OrderType()
```

Valeur de retour

Type de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE).

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TypeDescription

Retourne le type de l'ordre sous forme d'une chaîne de caractères.

```
string  TypeDescription() const
```

Valeur de retour

Type de l'ordre sous forme d'une chaîne de caractères.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Etat

Retourne l'état de l'ordre.

```
ENUM_ORDER_STATE  State() const
```

Valeur de retour

Etat de l'ordre (valeur de l'énumération ENUM_ORDER_STATE).

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

StateDescription

Retourne l'état de l'ordre sous forme d'une chaîne de caractères.

```
string  StateDescription() const
```

Valeur de retour

Etat de l'ordre sous forme d'une chaîne de caractères.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeExpiration

Retourne la date/heure d'expiration de l'ordre.

```
datetime  TimeExpiration() const
```

Valeur de retour

Date/heure d'expiration de l'ordre défini au moment du placement de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeDone

Retourne l'heure d'exécution ou d'annulation d'un ordre.

```
datetime  TimeDone() const
```

Valeur de retour

L'heure d'exécution ou d'annulation d'un ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeDoneMsc

Retourne l'heure d'exécution ou d'annulation d'un ordre en millisecondes depuis le 01/01/1970.

```
ulong  TimeDoneMsc() const
```

Valeur de retour

L'heure d'exécution ou d'annulation de l'ordre en millisecondes depuis le 01/01/1970.

Note

L'ordre doit tout d'abord être sélectionné pour pouvoir y accéder en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TypeFilling

Retourne le type de remplissage de l'ordre.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Valeur de retour

Type de remplissage de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_FILLING).

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TypeFillingDescription

Retourne le type de remplissage de l'ordre sous forme d'une chaîne de caractères.

```
string  TypeFillingDescription() const
```

Valeur de retour

Le type de remplissage de l'ordre sous forme d'une chaîne de caractères.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TypeTime

Retourne le type de l'ordre à l'heure d'expiration.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Valeur de retour

Le type de l'ordre àl'heure d'expiration.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TypeTimeDescription

Retourne le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères.

```
string  TypeTimeDescription() const
```

Valeur de retour

Le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Magic

Retourne l'identifiant de l'expert ayant placé l'ordre.

```
long  Magic() const
```

Valeur de retour

L'identifiant de l'expert ayant placé l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

PositionId

Retourne l'identifiant de la position.

```
long  PositionId() const
```

Valeur de retour

Identifiant de la position dans laquelle l'ordre est impliqué.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

VolumeInitial

Retourne le volume initial de l'ordre.

```
double  VolumeInitial() const
```

Valeur de retour

Le volume initial de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

VolumeCurrent

Retourne le volume non atteint de l'ordre.

```
double  VolumeCurrent() const
```

Valeur de retour

Le volume non atteint de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

PriceOpen

Retourne le prix de l'ordre.

```
double  PriceOpen() const
```

Valeur de retour

Prix de placement de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

StopLoss

Retourne le Stop Loss de l'ordre.

```
double  StopLoss() const
```

Valeur de retour

Le Stop Loss de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TakeProfit

Retourne le Take Profit de l'ordre.

```
double  TakeProfit() const
```

Valeur de retour

Le Take Profit de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

PriceCurrent

Retourne le prix courant du symbole

```
double  PriceCurrent() const
```

Valeur de retour

Le prix courant du symbole

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

PriceStopLimit

Retourne le prix de l'ordre limite.

```
double  PriceStopLimit() const
```

Valeur de retour

Le prix de l'ordre limite.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Symbol

Retourne le nom du symbole de l'ordre.

```
string  Symbol() const
```

Valeur de retour

Nom du symbole de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Comment

Retourne le commentaire de l'ordre

```
string  Comment() const
```

Valeur de retour

Commentaire de l'ordre.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,     // Identifiant de la propriété
   long&                        var          // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer (valeur de l'énumération ENUM_ORDER_PROPERTY_INTEGER).

var

[out] Référence à une variable de type long pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,     // Identifiant de la propriété
   double&                     var          // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double (valeur de l'énumération ENUM_ORDER_PROPERTY_DOUBLE).

var

[out] Référence à une variable de type double pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,     // Identifiant de la propriété
   string&                     var          // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété texte.

var

[out] Référence à une variable de type string pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre doit être sélectionné en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

StoreState

Sauvegarde les paramètres de l'ordre.

```
void  StoreState()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

CheckState

Vérifie les paramères actuels par rapport aux paramètres sauvegardés.

```
bool  CheckState()
```

Valeur de retour

vrai si les paramètres de l'ordre ont changé depuis le dernier appel à la méthode StoreState() , sinon faux.

© 2000-2015, MetaQuotes Software Corp.

Select

Sélectionne un ordre par son ticket pour accéder par la suite à ses propriétés.

```
bool  Select(
   ulong  ticket   // ticket de l'ordre
   )
```

Valeur de retour

vrai en cas de succès, faux si l'ordre n'a pas pu être sélectionné.

© 2000-2015, MetaQuotes Software Corp.

SelectByIndex

Sélectionne un ordre par son index pour accéder par la suite à ses propriétés.

```
bool  SelectByIndex(
   int  index   // index de l'ordre
   )
```

Valeur de retour

vrai en cas de succès, faux si l'ordre n'a pas pu être sélectionné.

© 2000-2015, MetaQuotes Software Corp.

CHistoryOrderInfo

La classe CHistoryOrderInfo permet d'accéder aux propriétés d'un ordre de l'historique.

Description

La classe CHistoryOrderInfo permet d'accéder aux propriétés d'un ordre de l'historique.

Déclaration

```
class CHistoryOrderInfo : public CObject
```

Titre

```
#include <Trade\HistoryOrderInfo.mqh>
```

Méthodes de classe par groupes

Accès aux propriétés de type integer	
TimeSetup	Retourne l'heure de placement d'un ordre
TimeSetupMsc	Retourne l'heure de placement d'un ordre en millisecondes depuis le 01/01/1970
OrderType	Retourne le type de l'ordre
OrderTypeDescription	Retourne le type de l'ordre sous forme d'une chaîne de caractères
Etat	Retourne l'état de l'ordre
StateDescription	Retourne l'état de l'ordre sous forme d'une chaîne de caractères
TimeExpiration	Retourne l'heure d'expiration d'un ordre
TimeDone	Retourne l'heure d'exécution ou d'annulation d'un ordre
TimeDoneMsc	Retourne l'heure d'exécution ou d'annulation d'un ordre en millisecondes depuis le 01/01/1970
TypeFilling	Retourne le type d'exécution de l'ordre par ordre
TypeFillingDescription	Retourne le type d'exécution de l'ordre par ordre sous forme d'une chaîne de caractères
TypeTime	Retourne le type de l'ordre à l'heure d'expiration
TypeTimeDescription	Retourne le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères
Magic	Retourne l'identifiant de l'expert ayant placé l'ordre

© 2000-2015, MetaQuotes Software Corp.

PositionId	Retourne l'identifiant de la position
Accès aux propriétés de type double	
VolumeInitial	Retourne le volume initial de l'ordre
VolumeCurrent	Retourne le volume non atteint de l'ordre
PriceOpen	Retourne le prix de l'ordre
StopLoss	Retourne le Stop Loss de l'ordre
TakeProfit	Retourne le Take Profit de l'ordre
PriceCurrent	Retourne le prix courant du symbole
PriceStopLimit	Retourne le prix de l'ordre limite
Accès aux propriétés de type texte	
Symbol	Retourne le symbole de l'ordre
Comment	Retourne le commentaire de l'ordre
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer
InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string
Sélection	
Ticket	Retourne le ticket/sélectionne l'ordre
SelectByIndex	Sélectionne l'ordre par son index

© 2000-2015, MetaQuotes Software Corp.

TimeSetup

Retourne l'heure de placement de l'ordre.

```
datetime  TimeSetup() const
```

Valeur de retour

L'heure de placement de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeSetupMsc

Retourne l'heure de placement d'un ordre en millisecondes depuis le 01/01/1970.

```
ulong  TimeSetupMsc() const
```

Valeur de retour

L'heure de placement d'un ordre en millisecondes depuis le 01/01/1970

Note

L'ordre historique doit tout d'abord être sélectionné pour pouvoir y accéder en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

OrderType

Retourne le type de l'ordre.

```
ENUM_ORDER_TYPE  OrderType() const
```

Valeur de retour

Type de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE).

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TypeDescription

Retourne le type de l'ordre sous forme d'une chaîne de caractères.

```
string  TypeDescription() const
```

Valeur de retour

Type de l'ordre sous forme d'une chaîne de caractères.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Etat

Retourne l'état de l'ordre.

```
ENUM_ORDER_STATE  State() const
```

Valeur de retour

Etat de l'ordre (valeur de l'énumération ENUM_ORDER_STATE).

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

StateDescription

Retourne l'état de l'ordre sous forme d'une chaîne de caractères.

```
string  StateDescription() const
```

Valeur de retour

Etat de l'ordre sous forme d'une chaîne de caractères.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeExpiration

Retourne l'heure d'expiration d'un ordre.

```
datetime  TimeExpiration() const
```

Valeur de retour

L'heure d'expiration d'un ordre, définit lors de son placement.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeDone

Retourne l'heure d'exécution ou d'annulation d'un ordre.

```
datetime  TimeDone() const
```

Valeur de retour

L'heure d'exécution ou d'annulation d'un ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeDoneMsc

Retourne l'heure d'exécution ou d'annulation d'un ordre en millisecondes depuis le 01/01/1970.

```
ulong  TimeDoneMsc() const
```

Valeur de retour

L'heure d'exécution ou d'annulation de l'ordre en millisecondes depuis le 01/01/1970.

Note

L'ordre historique doit tout d'abord être sélectionné pour pouvoir y accéder en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TypeFilling

Retourne le type d'exécution de l'ordre par ordre.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Valeur de retour

Type de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_FILLING).

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TypeFillingDescription

Retourne le type d'exécution de l'ordre par ordre sous forme d'une chaîne de caractères.

```
string  TypeFillingDescription() const
```

Valeur de retour

Le type d'exécution de l'ordre par ordre sous forme d'une chaîne de caractères.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TypeTime

Retourne le type de l'ordre à l'heure d'expiration.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Valeur de retour

Type de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TypeTimeDescription

Retourne le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères.

```
string  TypeTimeDescription() const
```

Valeur de retour

Le type de l'ordre à l'heure d'expiration sous forme d'une chaîne de caractères.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Magic

Retourne l'identifiant de l'Expert Advisor ayant placé l'ordre.

```
long  Magic() const
```

Valeur de retour

L'identifiant de l'Expert Advisor ayant placé l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

PositionId

Retourne l'identifiant de la position.

```
long  PositionId() const
```

Valeur de retour

Identifiant de la position dans laquelle l'ordre est impliqué.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

VolumeInitial

Retourne le volume initial de l'ordre.

```
double  VolumeInitial() const
```

Valeur de retour

Le volume initial de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

VolumeCurrent

Retourne le volume non atteint de l'ordre.

```
double  VolumeCurrent() const
```

Valeur de retour

Le volume non atteint de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

PriceOpen

Retourne le prix de l'ordre.

```
double  PriceOpen() const
```

Valeur de retour

Prix de placement de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

StopLoss

Retourne le prix du Stop Loss de l'ordre.

```
double  StopLoss() const
```

Valeur de retour

Le prix du Stop Loss de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TakeProfit

Retourne le prix du Take Profit de l'ordre.

```
double  TakeProfit() const
```

Valeur de retour

Le prix du Take Profit de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

PriceCurrent

Retourne le prix courant du symbole de l'ordre.

```
double  PriceCurrent() const
```

Valeur de retour

Le prix courant du symbole de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

PriceStopLimit

Retourne le prix stop limit de l'ordre.

```
double  PriceStopLimit() const
```

Valeur de retour

Prix stop limit de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Symbol

Retourne le nom du symbole de l'ordre.

```
string  Symbol() const
```

Valeur de retour

Nom du symbole de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Comment

Retourne le commentaire de l'ordre

```
string  Comment() const
```

Valeur de retour

Commentaire de l'ordre.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,     // Identifiant de la propriété
   long&                        var          // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer (valeur de l'énumération ENUM_ORDER_PROPERTY_INTEGER).

var

[out] Référence à une variable de type long pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,     // Identifiant de la propriété
   double&                     var          // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double (valeur de l'énumération ENUM_ORDER_PROPERTY_DOUBLE).

var

[out] Référence à une variable de type double pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,     // Identifiant de la propriété
   string&                     var          // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type texte (valeur de l'énumération ENUM_ORDER_PROPERTY_STRING).

var

[out] Référence à une variable de type string pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

L'ordre historique doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Ticket (Méthode "Get")

Retourne le ticket de l'ordre.

```
ulong  Ticket() const
```

Valeur de retour

Ticket de l'ordre.

Ticket (Méthode "Set")

Sélectionne l'ordre pour pouvoir l'utiliser ensuite.

```
void  Ticket(
   ulong  ticket      // ticket de l'ordre
   )
```

Paramètres

ticket

[in] Ticket de l'ordre.

© 2000-2015, MetaQuotes Software Corp.

SelectByIndex

Sélectionne un ordre par son index pour accéder par la suite à ses propriétés.

```
bool  SelectByIndex(
   int  index   // index de l'ordre
   )
```

Valeur de retour

vrai en cas de succès, faux si l'ordre n'a pas pu être sélectionné.

© 2000-2015, MetaQuotes Software Corp.

CPositionInfo

La classe CPositionInfo permet d'accéder aux propriétés des positions ouvertes.

Description

La classe CPositionInfo permet d'accéder aux propriétés des positions ouvertes.

Déclaration

```
class CPositionInfo : public CObject
```

Titre

```
#include <Trade\PositionInfo.mqh>
```

Méthodes de classe par groupes

Accès aux propriétés de type integer	
Time	Retourne l'heure de placement de l'ordre
TimeMsc	Retourne l'heure de placement de la position en millisecondes depuis le 01/01/1970
TimeUpdate	Retourne l'heure de changement de la position en secondes depuis le 01/01/1970
TimeUpdateMsc	Retourne l'heure de changement de la position en millisecondes depuis le 01/01/1970
PositionType	Retourne le type de la position
TypeDescription	Retourne le type de la position sous forme d'une chaîne de caractères
Magic	Retourne l'identifiant de l'expert ayant ouvert la position
Identifier	Retourne l'identifiant de la position
Accès aux propriétés de type double	
Volume	Retourne le volume de la position
PriceOpen	Retourne le prix d'ouverture de la position
StopLoss	Retourne le prix du Stop Loss de la position
TakeProfit	Retourne le prix du Take Profit de la position
PriceCurrent	Retourne le prix actuel du symbole de la position
Commission	Retourne le montant de la commission de la position
Swap	Retourne le montant du swap de la position

© 2000-2015, MetaQuotes Software Corp.

Profit	Retourne le montant du profit actuel de la position
Accès aux propriétés de type texte	
Symbol	Retourne le nom du symbole de la position
Comment	Retourne le commentaire de la position
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer
InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string.
Sélection	
Select	Sélectionne la position.
SelectByIndex	Sélectionne la position par son index
Etat	
StoreState	Sauvegarde les paramètres de la position
CheckState	Vérifie les paramères actuels par rapport aux paramètres sauvegardés

© 2000-2015, MetaQuotes Software Corp.

Time

Retourne l'heure d'ouverture de la position.

```
datetime  Time() const
```

Valeur de retour

L'heure d'ouverture de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeMsc

Retourne l'heure d'ouverture de la position en millisecondes depuis le 01/01/1970.

```
ulong  TimeMsc() const
```

Valeur de retour

L'heure d'ouverture de la position en millisecondes depuis le 01/01/1970.

Note

La position doit d'abord être sélectionnée en utilisant les méthodes Select (par le symbole) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeUpdate

Retourne l'heure de changement de la position en secondes depuis le 01/01/1970.

```
datetime  TimeUpdate() const
```

Valeur de retour

L'heure de changement de la position en secondes depuis le 01/01/1970.

Note

La position doit d'abord être sélectionnée en utilisant les méthodes Select (par le symbole) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TimeUpdateMsc

Retourne l'heure de changement de la position en millisecondes depuis le 01/01/1970.

```
ulong  TimeUpdateMsc() const
```

Valeur de retour

L'heure de changement de la position en millisecondes depuis le 01/01/1970.

Note

La position doit d'abord être sélectionnée en utilisant les méthodes Select (par le symbole) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

PositionType

Retourne le type de la position.

```
ENUM_POSITION_TYPE  PositionType() const
```

Valeur de retour

Type de la position (valeur de l'énumération ENUM_POSITION_TYPE).

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TypeDescription

Retourne le type de la position sous forme d'une chaîne de caractères.

```
string  TypeDescription() const
```

Valeur de retour

Le type de la position sous forme d'une chaîne de caractères.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Magic

Retourne l'identifiant de l'Expert Advisor ayant ouvert la position.

```
long  Magic() const
```

Valeur de retour

L'identifiant de l'Expert Advisor ayant ouvert la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Identifier

Retourne l'identifiant de la position.

```
long  Identifier() const
```

Valeur de retour

L'identifiant de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Volume

Retourne le volume de la position.

```
double  Volume() const
```

Valeur de retour

Le volume de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

PriceOpen

Retourne le prix d'ouverture de la position.

```
double  PriceOpen() const
```

Valeur de retour

Le prix d'ouverture de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

StopLoss

Retourne le prix du Stop Loss de la position.

```
double  StopLoss() const
```

Valeur de retour

Le prix du Stop Loss de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

TakeProfit

Retourne le prix du Take Profit de la position.

```
double  TakeProfit() const
```

Valeur de retour

Le prix du Take Profit de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

PriceCurrent

Retourne le prix actuel du symbole de la position.

```
double  PriceCurrent() const
```

Valeur de retour

Le prix actuel du symbole de la position.

© 2000-2015, MetaQuotes Software Corp.

Commission

Retourne le montant de la commission de la position.

```
double  Commission() const
```

Valeur de retour

Le montant de la commission de la position (dans la devise du compte).

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Swap

Retourne le montant du swap de la position.

```
double  Swap() const
```

Valeur de retour

Le montant du swap de la position (dans la devise du compte).

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Profit

Retourne le montant du profit actuel de la position.

```
double  Profit() const
```

Valeur de retour

Le montant du profit actuel de la position (dans la devise du compte).

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Symbol

Retourne le nom du symbole de la position.

```
string  Symbol() const
```

Valeur de retour

Le nom du symbole de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Comment

Retourne le commentaire de la position.

```
string  Comment() const
```

Valeur de retour

Le commentaire de la position.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
bool  InfoInteger(
   ENUM_POSITION_PROPERTY_INTEGER  prop_id,     // Identifiant de la propriété
   long&                           var          // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer (valeur de l'énumération ENUM_POSITION_PROPERTY_INTEGER).

var

[out] Référence à une variable de type long pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
bool  InfoDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  prop_id,      // Identifiant de la propriété
   double&                        var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double (valeur de l'énumération ENUM_POSITION_PROPERTY_DOUBLE).

var

[out] Référence à une variable de type double pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
bool  InfoString(
   ENUM_POSITION_PROPERTY_STRING  prop_id,      // Identifiant de la propriété
   string&                        var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type texte (valeur de l'énumération ENUM_POSITION_PROPERTY_STRING).

var

[out] Référence à une variable de type string pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

La position doit être sélectionnée en utilisant les méthodes Select (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Select

Sélectionne la position pour pouvoir l'utiliser ensuite.

```
bool  Select(
   const string  symbol      // symbole
   )
```

Paramètres

symbol

[in] Symbole pour la sélection de la position.

© 2000-2015, MetaQuotes Software Corp.

SelectByIndex

Sélectionne la position par son index pour accéder par la suite à ses propriétés.

```
bool  SelectByIndex(
   int  index   // index de la position
   )
```

Valeur de retour

vrai en cas de succès, faux si la position n'a pas pu être sélectionnée.

© 2000-2015, MetaQuotes Software Corp.

StoreState

Sauvegarde les paramètres de la position.

```
void  StoreState()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

CheckState

Vérifie les paramères actuels par rapport aux paramètres sauvegardés.

```
bool  CheckState()
```

Valeur de retour

vrai si les paramètres de la position ont changé depuis le dernier appel à la méthode StoreState() , sinon faux.

© 2000-2015, MetaQuotes Software Corp.

CDealInfo

La classe CDealInfo permet d'accéder aux propriétés du deal.

Description

La classe CDealInfo fournit l'accès aux propriétés du deal.

Déclaration

```
class CDealInfo : public CObject
```

Titre

```
#include <Trade\DealInfo.mqh>
```

Méthodes de classe par groupes

Accès aux propriétés de type integer	
Order	Retourne l'ordre correspondant au deal exécuté
Time	Retourne l'heure d'exécution du deal
TimeMsc	Retourne l'heure d'exécution d'un deal en millisecondes depuis le 01/01/1970
DealType	Retourne le type du deal
TypeDescription	Retourne le type du deal sous forme d'une chaîne de caractères
Entry	Retourne la direction du deal
EntryDescription	Retourne la direction du deal sous forme d'une chaîne de caractères
Magic	Retourne l'identifiant de l'expert ayant exécuté le deal
PositionId	Retourne l'identifiant de la position dans laquelle le deal est impliqué
Accès aux propriétés de type double	
Volume	Retourne le volume du deal
Price	Retourne le prix du deal
Commision	Retourne le montant de commission par deal
Swap	Retourne le montant du swap lorsque la position est clôturée
Profit	Retourne le résultat financier du deal
Accès aux propriétés de type texte	

© 2000-2015, MetaQuotes Software Corp.

Symbol	Retourne le nom du symbole du deal
Comment	Retourne le commentaire du deal
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété spécifiée de type integer
InfoDouble	Retourne la valeur de la propriété spécifiée de type double
InfoString	Retourne la valeur de la propriété spécifiée de type string
Sélection	
Ticket	Retourne le ticket/sélectionne le deal
SelectByIndex	Sélectionne le deal par son index

© 2000-2015, MetaQuotes Software Corp.

Order

Retourne l'ordre correspondant au deal exécuté.

```
long  Order() const
```

Valeur de retour

Ordre correspondant au deal exécuté.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

Time

Retourne l'heure d'exécution du deal.

```
datetime  Time() const
```

Valeur de retour

L'heure d'exécution du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

TimeMsc

Retourne l'heure d'exécution d'un deal en millisecondes depuis le 01/01/1970.

```
ulong  TimeMsc() const
```

Valeur de retour

L'heure d'exécution d'un deal en millisecondes depuis le 01/01/1970.

Note

Le deal doit tout d'abord être sélectionné pour pouvoir y accéder en utilisant les méthodes Ticket (par le ticket) ou SelectByIndex (par l'index).

© 2000-2015, MetaQuotes Software Corp.

DealType

Retourne le type du deal.

```
ENUM_DEAL_TYPE  DealType() const
```

Valeur de retour

Type du deal (valeur de l'énumération ENUM_DEAL_TYPE).

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

TypeDescription

Retourne le type du deal sous forme d'une chaîne de caractères.

```
string  TypeDescription() const
```

Valeur de retour

Type du deal sous forme d'une chaîne de caractères.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

Entry

Retourne la direction du deal.

```
ENUM_DEAL_ENTRY  Entry() const
```

Valeur de retour

Direction du deal (valeur de l'énumération ENUM_DEAL_ENTRY).

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

EntryDescription

Retourne la direction du deal sous forme d'une chaîne de caractères.

```
string  EntryDescription() const
```

Valeur de retour

Direction du deal sous forme d'une chaîne de caractères.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

Magic

Retourne l'identifiant de l'Expert Advisor ayant exécuté le deal.

```
long  Magic() const
```

Valeur de retour

Identifiant de l'Expert Advisor ayant exécuté le deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

PositionId

Retourne l'identifiant de la position dans laquelle le deal est impliqué.

```
long  PositionId() const
```

Valeur de retour

Identifiant de la position dans laquelle le deal est impliqué.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

Volume

Retourne le volume du deal.

```
double  Volume() const
```

Valeur de retour

Volume du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

Price

Retourne le prix du deal.

```
double  Price() const
```

Valeur de retour

Prix du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

Commission

Retourne le montant de commission du deal.

```
double  Commission() const
```

Valeur de retour

Le montant de commission du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

Swap

Retourne le montant du swap lorsque la position est clôturée.

```
double  Swap() const
```

Valeur de retour

Montant du swap lorsque la position est clôturée.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

Profit

Retourne le résultat financier du deal.

```
double  Profit() const
```

Valeur de retour

Résultat financier du deal (dans la devise du compte).

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

Symbol

Retourne le nom du symbole du deal.

```
string  Symbol() const
```

Valeur de retour

Le nom du symbole du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

Comment

Retourne le commentaire du deal.

```
string  Comment() const
```

Valeur de retour

Commentaire du deal.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

InfoInteger

Retourne la valeur de la propriété spécifiée de type integer.

```
bool  InfoInteger(
   ENUM_DEAL_PROPERTY_INTEGER  prop_id,     // Identifiant de la propriété
   long&                       var          // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type integer (valeur de l'énumération ENUM_DEAL_PROPERTY_INTEGER).

var

[out] Référence à une variable de type long pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

InfoDouble

Retourne la valeur de la propriété spécifiée de type double.

```
bool  InfoDouble(
   ENUM_DEAL_PROPERTY_DOUBLE  prop_id,      // Identifiant de la propriété
   double&                    var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type double (valeur de l'énumération ENUM_DEAL_PROPERTY_DOUBLE).

var

[out] Référence à une variable de type double pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

InfoString

Retourne la valeur de la propriété spécifiée de type string.

```
bool  InfoString(
   ENUM_DEAL_PROPERTY_STRING  prop_id,      // Identifiant de la propriété
   string&                    var           // référence sur la variable
   ) const
```

Paramètres

prop_id

[in] Identifiant de la propriété de type texte (valeur de l'énumération ENUM_DEAL_PROPERTY_STRING).

var

[out] Référence à une variable de type string pour placer le résultat.

Valeur de retour

vrai en cas de succès, faux si la valeur de la propriété n'a pas pu être récupérée.

Note

Le deal doit être sélectionné en utilisant les méthodes Ticket (par ticket) ou SelectByIndex (par index).

© 2000-2015, MetaQuotes Software Corp.

Ticket (Méthode "Get")

Retourne le ticket du deal.

```
ulong  Ticket() const
```

Valeur de retour

Ticket du deal.

Ticket (Méthode "Set")

Sélectionne la position pour pouvoir l'utiliser ensuite.

```
void  Ticket(
   ulong  ticket      // ticket
   )
```

Paramètres

ticket

[in] Ticket du deal.

© 2000-2015, MetaQuotes Software Corp.

SelectByIndex

Sélectionne le deal par son index pour accéder par la suite à ses propriétés.

```
bool  SelectByIndex(
   int  index   // index de l'ordre
   )
```

Valeur de retour

vrai en cas de succès, faux si le deal n'a pas pu être sélectionné.

© 2000-2015, MetaQuotes Software Corp.

CTrade

La classe CTrade est une classe permettant d'accéder aux fonctions de trading.

Description

La classe CTrade fournit l'accès aux fonctions de trading.

Déclaration

```
class CTrade : public CObject
```

Titre

```
#include <Trade\Trade.mqh>
```

Méthodes de classe par groupes

Définition des paramètres	
LogLevel	Définit le niveau de journalisation
SetExpertMagicNumber	Définit l'identifiant de l'expert
SetDeviationInPoints	Définit la déviation autorisée
SetTypeFilling	Définit le type de remplissage de l'ordre
SetAsyncMode	Définit le mode asynchrone pour les opérations de trading
Opérations sur les ordres	
OrderOpen	Place un ordre en attente avec les paramètres définis
OrderModify	Modifie les paramètres d'un ordre en attente
OrderDelete	Supprime un ordre en attente
Opérations sur les positions	
PositionOpen	Ouvre une position avec les paramètres définis
PositionModify	Modifie les paramètres d'une position
PositionClose	Clôture une position
Méthodes supplémentaires	
Buy	Ouvre une position longue avec les paramètres spécifiés
Sell	Ouvre une position courte avec les paramètres spécifiés
BuyLimit	Place un ordre en attente de type Buy Limit avec les paramètres spécifiés

© 2000-2015, MetaQuotes Software Corp.

BuyStop	Place un ordre en attente de type Buy Stop avec les paramètres spécifiés
SellLimit	Place un ordre en attente de type Sell Limit avec les paramètres spécifiés
SellStop	Place un ordre en attente de type Sell Stop avec les paramètres spécifiés
Accès aux paramètres de la dernière requête	
Request	Retourne une copie de la structure de la dernière requête
RequestAction	Retourne le type d'une opération de trading
RequestActionDescription	Retourne le type d'une opération de trading sous forme d'une chaîne de caractères
RequestMagic	Retourne l'identifiant Magic de l'Expert Advisor
RequestOrder	Retourne le ticket de l'ordre utilisé dans la dernière requête
RequestSymbol	Retourne le nom du symbole utilisé dans la dernière requête
RequestVolume	Retourne le volume (en lots) utilisé dans la dernière requête
RequestPrice	Retourne le prix utilisé dans la dernière requête
RequestStopLimit	Retourne le prix d'un ordre en attente de type Stop Limit, utilisé dans la dernière requête
RequestSL	Retourne le prix du Stop Loss de l'ordre, utilisé dans la dernière requête
RequestTP	Retourne le prix du Take Profit de l'ordre, utilisé dans la dernière requête
RequestDeviation	Retourne la déviation du prix de l'ordre, utilisé dans la dernière requête
RequestType	Retourne le type de l'ordre, utilisé dans la dernière requête
RequestTypeDescription	Retourne le type de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête
RequestTypeFilling	Retourne le type de remplissage de l'ordre, utilisé dans la dernière requête
RequestTypeFillingDescription	Retourne le type de remplissage de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête
RequestTypeTime	Retourne la période de validité de l'ordre,

© 2000-2015, MetaQuotes Software Corp.

	utilisé dans la dernière requête
RequestTypeTimeDescription	Retourne la période de validité de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête
RequestExpiration	Retourne l'heure d'expiration de l'ordre utilisé dans la dernière requête
RequestComment	Retourne le commentaire de l'ordre utilisé dans la dernière requête
Accès aux résultat de la dernière requête	
CheckResult	Retourne une copie de la structure des résultats de la dernière requête.
CheckResultRetcode	Retourne la valeur du champ retcode du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête
CheckResultRetcodeDescription	Retourne la description textuelle du champ retcode du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête
CheckResultBalance	Retourne la valeur du champ balance du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête
CheckResultEquity	Retourne la valeur du champ equity du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête
CheckResultProfit	Retourne la valeur du champ profit du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête
CheckResultMargin	Retourne la valeur du champ margin du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête
CheckResultMarginFree	Retourne la valeur du champ margin_free du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête
CheckResultMarginLevel	Retourne la valeur du champ margin_level du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête
CheckResultComment	Retourne la valeur du champ comment du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête
Accès aux résultat de l'exécution de la dernière requête	

© 2000-2015, MetaQuotes Software Corp.

Result	Retourne une copie de la structure des résultats de la dernière requête.
ResultRetcode	Retourne le code du résultat de la requête
ResultRetcodeDescription	Retourne le code du résultat de la requête sous forme de texte
ResultDeal	Retourne le ticket du deal
ResultOrder	Retourne le ticket de l'ordre
ResultVolume	Retourne le volume du deal ou de l'ordre
ResultPrice	Retourne le prix confirmé par le courtier
ResultBid	Retourne le prix bid courant (la recotation)
ResultAsk	Retourne le prix ask courant (la recotation)
ResultComment	Retourne le commentaire du courtier
Autres méthodes	
PrintRequest	Ecrit les paramètres de la dernière requête dans le journal
PrintResult	Ecrit les résultats de la dernière requête dans le journal
FormatRequest	Prépare la chaîne de caractères formatée avec les paramètres de la dernière requête
FormatRequestResult	Prépare la chaîne de caractères formatée avec les résultats d'exécution de la dernière requête

© 2000-2015, MetaQuotes Software Corp.

LogLevel

Définit le niveau de journalisation des messages.

```
void  LogLevel(
   ENUM_LOG_LEVELS  log_level      // niveau de journalisation
   )
```

Paramètres

log_level

[in] Niveau de journalisation.

Valeur de retour

Aucune.

Note

LOG_LEVEL_NO et en-dessous désactive l'affichage de tous les messages (définit automatiquement dans le mode d'optimisation). LOG_LEVEL_ERRORS active l'affichage des messages d'erreurs uniquement (valeur par défaut). LOG_LEVEL_ALL et au-dessus active l'affichage de tous les messages (définit automatiquement dans le mode de test).

ENUM_LOG_LEVELS

Identifier	Description	Value
LOG_LEVEL_NO	L'affichage des messages est désactivé	0
LOG_LEVEL_ERRORS	Seuls les messages d'erreurs sont affichés	1
LOG_LEVEL_ALL	Tous les messages sont affichés	2

© 2000-2015, MetaQuotes Software Corp.

SetExpertMagicNumber

Définit l'identifiant de l'expert.

```
void  SetExpertMagicNumber(
   ulong  magic      // Identifiant
   )
```

Paramètres

magic

[in] Nouvel identifiant de l'expert.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

SetDeviationInPoints

Définit la déviation autorisée.

```
void  SetDeviationInPoints(
   ulong  deviation      // Déviation
   )
```

Paramètres

deviation

[in] Déviation autorisée.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

SetTypeFilling

Définit le type de remplissage de l'ordre.

```
void  SetTypeFilling(
   ENUM_ORDER_TYPE_FILLING  filling      // type de remplissage de l'ordre
   )
```

Paramètres

filling

[in] Type de remplissage de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_FILLING).

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

SetAsyncMode

Définit le mode asynchrone pour les opérations de trading.

```
void  SetAsyncMode(
   bool  mode      // flag du mode asynchrone
  )
```

Paramètres

mode

[in] Flag du mode asynchrone.

Valeur de retour

Aucune.

Note

Ce mode est utilisé pour les opérations de trading en mode asynchrone (sans attendre la réponse du serveur de trading à la requête envoyée) - voir OrderSendAsync.

© 2000-2015, MetaQuotes Software Corp.

OrderOpen

Place un ordre en attente avec les paramètres définis.

```
bool  OrderOpen(
   const string          symbol,          // symbole
   ENUM_ORDER_TYPE       order_type,      // type d'ordre
   double                volume,          // volume de l'ordre
   double                limit_price,     // prix StopLimit
   double                price,           // prix d'exécution
   double                sl,              // prix du stop loss
   double                tp,              // prix du take profit
   ENUM_ORDER_TYPE_TIME  type_time,       // type par expiration
   datetime              expiration,      // expiration
   const string          comment=""       // commentaire
   )
```

Paramètres

symbol

[in] Nom de l'instrument de trading.

order_type

[in] Type de l'opération de trading (valeur de l'énumération ENUM_ORDER_TYPE).

volume

[in] Volume demandé pour l'ordre.

limit_price

[in] Prix auquel l'ordre StopLimit sera placé.

price

[in] Prix auquel l'ordre doit être exécuté.

sl

[in] Prix auquel le Stop Loss sera déclenché.

tp

[in] Prix auquel le Take Profit sera déclenché.

type_time

[in] Type de validité de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

expiration

[in] Date d'expiration de l'ordre en attente.

comment=""

[in] Commentaire de l'ordre.

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

Note

La réussite de la méthode OrderSend(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par <li5>ResultOrder()</li5><t6>.</t6>******

© 2000-2015, MetaQuotes Software Corp.

OrderModify

Modifie les paramètres d'un ordre en attente.

```
bool  OrderModify(
   ulong                ticket,        // ticket de l'ordre
   double               price,         // prix d'exécution
   double               sl,            // prix du stop loss
   double               tp,            // prix du take profit
   ENUM_ORDER_TYPE_TIME type_time,     // type par expiration
   datetime             expiration,    // expiration
   double               stoplimit      // Prix limite de l'ordre
   )
```

Paramètres

ticket

[in] Ticket de l'ordre.

price

[in] Le nouveau prix auquel l'ordre doit être exécuté (ou la valeur précédente si le changement n'est pas nécessaire).

sl

[in] Le nouveau prix auquel le Stop Loss sera déclenché (ou la valeur précédente si le changement n'est pas nécessaire).

tp

[in] Le nouveau prix auquel le Take Profit sera déclenché (ou la valeur précédente si le changement n'est pas nécessaire).

type_time

[in] Le nouveau type d'expiration de l'ordre (ou la valeur précédente si le changement n'est pas nécessaire), valeur de l'énumération ENUM_ORDER_TYPE_TIME.

expiration

[in] La nouvelle date d'expiration de l'ordre en attente (ou la valeur précédente si le changement n'est pas nécessaire).

stoplimit

[in] Le nouveau prix utilisé pour placer un ordre Limit lorsque le prix atteint la valeur du paramètre *price* . Il n'est spécifié que pour les ordres StopLimit.

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

Note

La réussite de la méthode OrderModify(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode().

© 2000-2015, MetaQuotes Software Corp.

OrderDelete

Supprime un ordre en attente.

```
bool  OrderDelete(
   ulong  ticket      // ticket de l'ordre
   )
```

Paramètres

ticket

[in] Ticket de l'ordre.

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

Note

La réussite de la méthode OrderDelete(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode().

© 2000-2015, MetaQuotes Software Corp.

PositionOpen

Ouvre une position avec les paramètres définis.

```
bool  PositionOpen(
   const string      symbol,          // symbole
   ENUM_ORDER_TYPE   order_type,      // type d'ordre to open position
   double            volume,          // volume de la position
   double            price,           // prix d'exécution
   double            sl,              // prix du stop loss
   double            tp,              // prix du take profit
   const string      comment=""       // commentaire
   )
```

Paramètres

symbol

[in] Nom de l'instrument de trading sur lequel la position doit être ouverte.

order_type

[in] Type d'ordre (opération de trading) pour ouvrir une position (valeur de l'énumération ENUM_ORDER_TYPE).

volume

[in] Volume demandé pour la position.

price

[in] Prix auquel la position doit être ouverte.

sl

[in] Prix auquel le Stop Loss sera déclenché.

tp

[in] Prix auquel le Take Profit sera déclenché.

comment=""

[in] Commentaire de la position.

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

Note

La réussite de la méthode PositionOpen(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par <li5>ResultDeal()</li5><t6>.</t6>

© 2000-2015, MetaQuotes Software Corp.

PositionModify

Modifie les paramètres d'une position sur le symbole spécifié.

```
bool  PositionModify(
   const string  symbol,      // symbole
   double        sl,          // prix du stop loss
   double        tp           // prix du take profit
   )
```

Paramètres

symbol

[in] Nom de l'instrument de trading sur lequel la position doit être modifiée.

sl

[in] Le nouveau prix auquel le Stop Loss sera déclenché (ou la valeur précédente si le changement n'est pas nécessaire).

tp

[in] Le nouveau prix auquel le Take Profit sera déclenché (ou la valeur précédente si le changement n'est pas nécessaire).

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

Note

La réussite de la méthode PositionModify(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode().

© 2000-2015, MetaQuotes Software Corp.

PositionClose

Clôture une position sur le symbole spécifié.

```
bool  PositionClose(
   const string  symbol,                  // symbole
   ulong         deviation=ULONG_MAX      // Déviation
   )
```

Paramètres

symbol

[in] Nom de l'instrument de trading sur lequel la position doit être clôturée.

deviation=ULONG_MAX

[in] Déviation maximale par rapport au prix courant (en points).

Valeur de retour

vrai si la vérification des structures de base est correcte, faux sinon.

Note

La réussite de la méthode PositionClose(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode().

© 2000-2015, MetaQuotes Software Corp.

Buy

Ouvre une position longue avec les paramètres spécifiés.

```
bool  Buy(
   double         volume,          // volume de la position
   const string   symbol=NULL,     // symbole
   double         price=0.0,       // prix
   double         sl=0.0,          // prix du stop loss
   double         tp=0.0,          // prix du take profit
   const string   comment=""       // commentaire
   )
```

Paramètres

volume

[in] Volume de la position.

symbol=NULL

[in] Symbole de la position. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

price=0.0

[in] Prix. Si le prix n'est pas spécifié, le prix Ask du marché courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

comment=""

[in] Commentaire.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode Buy(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultDeal().

© 2000-2015, MetaQuotes Software Corp.

Sell

Ouvre une position courte avec les paramètres spécifiés.

```
bool  Sell(
   double         volume,          // volume de la position
   const string   symbol=NULL,     // symbole
   double         price=0.0,       // prix
   double         sl=0.0,          // prix du stop loss
   double         tp=0.0,          // prix du take profit
   const string   comment=""       // commentaire
   )
```

Paramètres

volume

[in] Volume de la position.

symbol=NULL

[in] Symbole de la position. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

price=0.0

[in] Prix. Si le prix n'est pas spécifié, le prix Bid du marché courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

comment=""

[in] Commentaire.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode Sell(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultDeal().

© 2000-2015, MetaQuotes Software Corp.

BuyLimit

Place un ordre en attente de type Buy Limit (achat au prix, plus bas que le prix actuel du marché) avec les paramètres spécifiés.

```
bool  BuyLimit(
   double                volume,                    // volume de l'ordre
   double                price,                     // prix de l'ordre
   const string          symbol=NULL,               // symbole
   double                sl=0.0,                    // prix du stop loss
   double                tp=0.0,                    // prix du take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // durée de vie de l'ordre
   datetime              expiration=0,              // heure d'expiration de l'ordr
   const string          comment=""                 // commentaire
   )
```

Paramètres

volume

[in] Volume de l'ordre.

price

[in] Prix de l'ordre.

symbol=NULL

[in] Symbole de l'ordre. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

type_time=ORDER_TIME_GTC

[in] Durée de vie de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Heure d'expiration de l'ordre (utilisé uniquement si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commentaire de l'ordre.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode BuyLimit(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultDeal().

© 2000-2015, MetaQuotes Software Corp.

BuyStop

Place un ordre en attente de type Buy Stop (achat au prix, plus haut que le prix actuel du marché) avec les paramètres spécifiés.

```
bool  BuyStop(
   double                volume,                    // volume de l'ordre
   double                price,                     // prix de l'ordre
   const string          symbol=NULL,               // symbole
   double                sl=0.0,                    // prix du stop loss
   double                tp=0.0,                    // prix du take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // durée de vie de l'ordre
   datetime              expiration=0,              // heure d'expiration de l'ordr
   const string          comment=""                 // commentaire
   )
```

Paramètres

volume

[in] Volume de l'ordre.

price

[in] Prix de l'ordre.

symbol=NULL

[in] Symbole de l'ordre. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

type_time=ORDER_TIME_GTC

[in] Durée de vie de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Heure d'expiration de l'ordre (utilisé uniquement si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commentaire de l'ordre.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode BuyStop(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultDeal().

© 2000-2015, MetaQuotes Software Corp.

SellLimit

Place un ordre en attente de type Sell Limit (vente au prix, plus haut que le prix actuel du marché) avec les paramètres spécifiés.

```
bool  SellLimit(
   double                volume,                    // volume de l'ordre
   double                price,                     // prix de l'ordre
   const string          symbol=NULL,               // symbole
   double                sl=0.0,                    // prix du stop loss
   double                tp=0.0,                    // prix du take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // durée de vie de l'ordre
   datetime              expiration=0,              // heure d'expiration de l'ordr
   const string          comment=""                 // commentaire
   )
```

Paramètres

volume

[in] Volume de l'ordre.

price

[in] Prix de l'ordre.

symbol=NULL

[in] Symbole de l'ordre. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

type_time=ORDER_TIME_GTC

[in] Durée de vie de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Heure d'expiration de l'ordre (utilisé uniquement si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commentaire de l'ordre.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode SellLimit(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultDeal().

© 2000-2015, MetaQuotes Software Corp.

SellStop

Place un ordre en attente de type Sell Stop (vente au prix, plus bas que le prix actuel du marché) avec les paramètres spécifiés.

```
bool  SellStop(
   double                volume,                    // volume de l'ordre
   double                price,                     // prix de l'ordre
   const string          symbol=NULL,               // symbole
   double                sl=0.0,                    // prix du stop loss
   double                tp=0.0,                    // prix du take profit
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // durée de vie de l'ordre
   datetime              expiration=0,              // heure d'expiration de l'ordr
   const string          comment=""                 // commentaire
   )
```

Paramètres

volume

[in] Volume de l'ordre.

price

[in] Prix de l'ordre.

symbol=NULL

[in] Symbole de l'ordre. Si le symbole n'est pas spécifié, le symbole courant est utilisé.

sl=0.0

[in] Prix du Stop Loss

tp=0.0

[in] Prix du Take Profit

type_time=ORDER_TIME_GTC

[in] Durée de vie de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Heure d'expiration de l'ordre (utilisé uniquement si type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Commentaire de l'ordre.

Valeur de retour

vrai si la vérification des structures est correcte, faux sinon.

Note

La réussite de la méthode SellStop(...) ne signifie pas forcément une exécution avec succès de l'opération de trading. Il est nécessaire de vérifier le résultat de la requête de trading (le code de retour du serveur de trading) en utilisant ResultRetcode() et la valeur, retournée par ResultDeal().

© 2000-2015, MetaQuotes Software Corp.

Request

Retourne une copie de la structure de la dernière requête.

```
void  Request(
   MqlTradeRequest&  request      // structure destination
   ) const
```

Paramètres

request

[out] Référence à une structure de type MqlTradeRequest.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

RequestAction

Retourne le type d'une opération de trading.

```
ENUM_TRADE_REQUEST_ACTIONS  RequestAction() const
```

Valeur de retour

Type de l'opération de trading, utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestActionDescription

Retourne le type d'une opération de trading sous forme d'une chaîne de caractères.

```
string  RequestActionDescription() const
```

Valeur de retour

Type de l'opération de trading (sous forme d'une chaîne de caractères), utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestMagic

Retourne l'identifiant Magic de l'Expert Advisor.

```
ulong  RequestMagic() const
```

Valeur de retour

L'identifiant Magic de l'Expert Advisor, utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestOrder

Retourne le ticket de l'ordre utilisé dans la dernière requête.

```
ulong  RequestOrder() const
```

Valeur de retour

Le ticket de l'ordre de la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestSymbol

Retourne le nom du symbole utilisé dans la dernière requête.

```
string  RequestSymbol() const
```

Valeur de retour

Le nom du symbole utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestVolume

Retourne le volume de trading (en lots) utilisé dans la dernière requête.

```
double  RequestVolume() const
```

Valeur de retour

Le volume de trading (en lots), utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestPrice

Retourne le prix utilisé dans la dernière requête.

```
double  RequestPrice() const
```

Valeur de retour

Le prix de l'ordre utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestStopLimit

Retourne le prix d'un ordre en attente de type Stop Limit, utilisé dans la dernière requête.

```
double  RequestStoplimit() const
```

Valeur de retour

Le prix de l'ordre en attente de type Stop Limit, utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestSL

Retourne le prix du Stop Loss de l'ordre, utilisé dans la dernière requête.

```
double  RequestSL() const
```

Valeur de retour

Le prix du Stop Loss, utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestTP

Retourne le prix du Take Profit de l'ordre, utilisé dans la dernière requête.

```
double  RequestTP() const
```

Valeur de retour

Le prix du Take Profit , utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestDeviation

Retourne la déviation du prix de l'ordre, utilisé dans la dernière requête.

```
ulong  RequestDeviation() const
```

Valeur de retour

La déviation du prix de l'ordre, utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestType

Retourne le type de l'ordre, utilisé dans la dernière requête.

```
ENUM_ORDER_TYPE  RequestType() const
```

Valeur de retour

Le type de l'ordre utilisé dans la dernière requête (valeur de l'énumération ENUM_ORDER_TYPE).

© 2000-2015, MetaQuotes Software Corp.

RequestTypeDescription

Retourne le type de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête.

```
string  RequestTypeDescription() const
```

Valeur de retour

Le type de l'ordre (sous forme d'une chaîne de caractères) utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestTypeFilling

Retourne le type de remplissage de l'ordre, utilisé dans la dernière requête.

```
ENUM_ORDER_TYPE_FILLING  RequestTypeFilling() const
```

Valeur de retour

Le type de remplissage de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_FILLING), utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestTypeFillingDescription

Retourne le type de remplissage de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête.

```
string  RequestTypeFillingDescription() const
```

Valeur de retour

Le type de remplissage (sous forme d'une chaîne de caractères) de l'ordre, utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestTypeTime

Retourne la période de validité de l'ordre, utilisé dans la dernière requête.

```
ENUM_ORDER_TYPE_TIME  RequestTypeTime() const
```

Valeur de retour

La période de validité de l'ordre (valeur de l'énumération ENUM_ORDER_TYPE_TIME), utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestTypeTimeDescription

Retourne la période de validité de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête.

```
string  RequestTypeTimeDescription() const
```

Valeur de retour

La période de validité de l'ordre (sous forme d'une chaîne de caractères), utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestExpiration

Retourne l'heure d'expiration de l'ordre utilisé dans la dernière requête.

```
datetime  RequestExpiration() const
```

Valeur de retour

L'heure d'expiration de l'ordre utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

RequestComment

Retourne le commentaire de l'ordre utilisé dans la dernière requête.

```
string  RequestComment() const
```

Valeur de retour

Le commentaire de l'ordre utilisé dans la dernière requête.

© 2000-2015, MetaQuotes Software Corp.

Result

Retourne une copie de la structure des résultats de la dernière requête.

```
void  Result(
   MqlTradeResult&  result      // référence
   ) const
```

Paramètres

result

[out] Référence à une structure de type MqlTradeRequest.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

ResultRetcode

Retourne le code du résultat de la requête.

```
uint  ResultRetcode() const
```

Valeur de retour

Le Code du résultat de la requête.

© 2000-2015, MetaQuotes Software Corp.

ResultRetcodeDescription

Retourne le code du résultat de la requête sous forme de texte.

```
string  ResultRetcodeDescription() const
```

Valeur de retour

Le code du résultat de la dernière requête sous forme de texte.

© 2000-2015, MetaQuotes Software Corp.

ResultDeal

Retourne le ticket du deal.

```
ulong  ResultDeal() const
```

Valeur de retour

Le ticket du deal, si le deal est exécuté.

© 2000-2015, MetaQuotes Software Corp.

ResultOrder

Retourne le ticket de l'ordre.

```
ulong  ResultOrder() const
```

Valeur de retour

Le ticket de l'ordre, si l'ordre est placé.

© 2000-2015, MetaQuotes Software Corp.

ResultVolume

Retourne le volume du deal ou de l'ordre.

```
double  ResultVolume() const
```

Valeur de retour

Le volume du deal ou de l'ordre.

© 2000-2015, MetaQuotes Software Corp.

ResultPrice

Retourne le prix confirmé par le courtier.

```
double  ResultPrice() const
```

Valeur de retour

Le prix, confirmé par le courtier.

© 2000-2015, MetaQuotes Software Corp.

ResultBid

Retourne le prix bid courant (la recotation).

```
double  ResultBid() const
```

Valeur de retour

Le prix bid courant (la recotation).

© 2000-2015, MetaQuotes Software Corp.

ResultAsk

Retourne le prix ask courant (la recotation).

```
double  ResultAsk() const
```

Valeur de retour

Le prix ask courant (la recotation).

© 2000-2015, MetaQuotes Software Corp.

ResultComment

Retourne le commentaire du courtier.

```
string  ResultComment() const
```

Valeur de retour

Le commentaire du courtier pour l'opération.

© 2000-2015, MetaQuotes Software Corp.

CheckResult

Retourne une copie de la structure des résultats de la dernière requête.

```
void  CheckResult(
   MqlTradeCheckResult&  check_result      // référence
   ) const
```

Paramètres

check_result

[out] Référence à la structure destination de type MqlTradeCheckResult.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

CheckResultRetcode

Retourne la valeur du champ retcode du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
uint  CheckResultRetcode() const
```

Valeur de retour

La valeur du champ retcode (code d'erreur) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2015, MetaQuotes Software Corp.

CheckResultRetcodeDescription

Retourne la description textuelle du champ retcode du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
string  ResultRetcodeDescription() const
```

Valeur de retour

La description textuelle du champ retcode (code d'erreur) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2015, MetaQuotes Software Corp.

CheckResultBalance

Retourne la valeur du champ balance du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double  CheckResultBalance() const
```

Valeur de retour

La valeur du champ balance (la valeur de la balance qui résultera de l'exécution de l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2015, MetaQuotes Software Corp.

CheckResultEquity

Retourne la valeur du champ equity du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double  CheckResultEquity() const
```

Valeur de retour

La valeur du champ equity (la valeur du capital qui résultera de l'exécution de l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2015, MetaQuotes Software Corp.

CheckResultProfit

Retourne la valeur du champ profit du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double  CheckResultProfit() const
```

Valeur de retour

La valeur du champ profit (la valeur du profit qui résultera de l'exécution de l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2015, MetaQuotes Software Corp.

CheckResultMargin

Retourne la valeur du champ margin du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double  CheckResultMargin() const
```

Valeur de retour

La valeur du champ margin (la marge requise pour l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2015, MetaQuotes Software Corp.

CheckResultMarginFree

Retourne la valeur du champ margin_free du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double  CheckResultMarginFree() const
```

Valeur de retour

La valeur du champ margin_free (la marge libre qui résultera de l'exécution de l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2015, MetaQuotes Software Corp.

CheckResultMarginLevel

Retourne la valeur du champ margin_level du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
double  CheckResultMarginLevel() const
```

Valeur de retour

La valeur du champ margin_level (le niveau de marge qui résultera de l'exécution de l'opération de trading) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2015, MetaQuotes Software Corp.

CheckResultComment

Retourne la valeur du champ comment du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

```
string  CheckResultComment() const
```

Valeur de retour

La valeur du champ comment (commentaire de code de retour, description de l'erreur) du type MqlTradeCheckResult, remplit lors de la vérification de l'exactitude de la requête.

© 2000-2015, MetaQuotes Software Corp.

PrintRequest

Ecrit les paramètres de la dernière requête dans le journal.

```
void  PrintRequest() const
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

PrintResult

Ecrit les résultats de la dernière requête dans le journal.

```
void  PrintResult() const
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

FormatRequest

Prépare la chaîne de caractères formatée avec les paramètres de la dernière requête.

```
string  FormatRequest(
   string&                 str,          // chaîne de caractères destination
   const MqlTradeRequest&  request       // requête
   ) const
```

Paramètres

str

[in] Chaîne de caractères destination, passée par référence.

request

[in] Une structure de type MqlTradeRequest avec les paramètres de la dernière requête.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

FormatRequestResult

Prépare la chaîne de caractères formatée avec les résultats d'exécution de la dernière requête.

```
string  FormatRequestResult(
   string&                 str,          // chaîne de caractères
   const MqlTradeRequest&  request,      // structure de la requête
   const MqlTradeResult&   result        // structure de résultat
   ) const
```

Paramètres

str

[in] Chaîne de caractères destination, passée par référence.

request

[in] Une structure de type MqlTradeRequest avec les paramètres de la dernière requête.

result

[in] Une structure de type MqlTradeResult avec les résultats de la dernière requête.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

CTerminalInfo

CString est une classe permettant un accès simplifié aux propriétés de l'environnement du programme MQL5.

Description

La classe CTerminalInfo permet d'accéder aux propriétés de l'environnement du programme MQL5.

Déclaration

```
class CTerminalInfo : public CObject
```

Titre

```
#include <Trade\TerminalInfo.mqh>
```

Méthodes de classe par groupes

Méthodes d'accès aux propriétés de type integer	
Build	Retourne le numéro de build du terminal client
IsConnected	Retourne les informations de connexion au serveur de trading
IsDLLsAllowed	Retourne l'autorisation d'utiliser des DLL
IsTradeAllowed	Retourne l'autorisation d'effectuer des opération de trading
IsEmailEnabled	Retourne l'autorisation d'envoyer des emails au serveur/login SMTP spécifié dans les paramètres du terminal
IsFtpEnabled	Retourne l'autorisation d'envoyer des rapports de trading au serveur/login FTP spécifié dans les paramètres du terminal
MaxBars	Retourne le nombre maximum de barres du graphique
CodePage	Retourne la page de code de la langue du terminal client
CPUCores	Retourne le nombre de coeurs CPU
MemoryPhysical	Retourne la quantité de mémoire physique (en Mo)
MemoryTotal	Retourne la quantité de mémoire totale disponible pour le processus de l'agent ou du terminal (en Mo)
MemoryAvailable	Retourne la quantité de mémoire libre,

© 2000-2015, MetaQuotes Software Corp.

	disponible pour pour le processus de l'agent ou du terminal (en Mo)
MemoryUsed	Retourne la quantité de mémoire utilisée par le processus de l'agent ou du terminal (en Mo)
IsX64	Retourne le type du terminal client (32/64 bits)
OpenCLSupport	Retourne la version de OpenCL supportée par la carte vidéo
DiskSpace	Retourne la quantité d'espace disque disponible (en Mo)
Méthodes d'accès aux propriétés de type string	
Language	Retourne la langue du terminal client
Name	Retourne le nom du terminal client
Company	Retourne la société du terminal client
Path	Retourne le répertoire du terminal client
DataPath	Retourne le répertoire des données du terminal client
CommonDataPath	Retourne le répertoire des données communes de tous les terminaux client installés sur l'ordinateur
Accès aux fonctions de l'API MQL5	
InfoInteger	Retourne la valeur de la propriété de type integer
InfoString	Retourne la valeur de la propriété de type string

© 2000-2015, MetaQuotes Software Corp.

Build

Retourne le numéro de build du terminal client.

```
int  CBuild() const
```

Valeur de retour

Le numéro de build du terminal client.

Note

Pour récupérer le numéro de build, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_BUILD).

© 2000-2015, MetaQuotes Software Corp.

IsConnected

Retourne les informations de connexion au serveur de trading.

```
bool  IsConnected() const
```

Valeur de retour

vrai si le terminal est connecté au serveur de trading, faux sinon.

Note

Pour récupérer le statut de la connexion, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_CONNECTED).

© 2000-2015, MetaQuotes Software Corp.

IsDLLsAllowed

Retourne l'autorisation d'utiliser des DLL.

```
bool  IsDLLsAllowed() const
```

Valeur de retour

vrai si l'utilisation des DLL est autorisée, faux sinon.

Note

Pour récupérer l'autorisation d'utiliser les DLL, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_DLLS_ALLOWED).

© 2000-2015, MetaQuotes Software Corp.

IsTradeAllowed

Retourne l'autorisation d'effectuer des opération de trading.

```
bool  IsTradeAllowed() const
```

Valeur de retour

vrai si le trading est autorisé, faux sinon.

Note

Pour récupérer l'autorisation d'effectuer des opérations de trading, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_TRADE_ALLOWED).

© 2000-2015, MetaQuotes Software Corp.

IsEmailEnabled

Retourne l'autorisation d'envoyer des emails au serveur/login SMTP spécifié dans les paramètres du terminal.

```
bool  IsEmailEnabled() const
```

Valeur de retour

vrai si l'envoi d'emails est autorisé, faux sinon.

Note

Pour récupérer l'autorisation d'envoyer des emails, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_EMAIL_ENABLED).

© 2000-2015, MetaQuotes Software Corp.

IsFtpEnabled

Retourne l'autorisation d'envoyer des rapports de trading au serveur/login FTP spécifié dans les paramètres du terminal.

```
bool  IsFtpEnabled() const
```

Valeur de retour

vrai si l'envoi de rapports de trading au serveur FTP est autorisé, faux sinon.

Note

Pour récupérer l'autorisation d'envoyer des rapports de trading, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_FTP_ENABLED).

© 2000-2015, MetaQuotes Software Corp.

MaxBars

Retourne le nombre maximum de barres du graphique spécifié dans les paramètres du terminal client.

```
int  MaxBars() const
```

Valeur de retour

Le nombre maximum de barres du graphique.

Note

Pour récupérer le nombre de barres du graphique, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_MAXBARS).

© 2000-2015, MetaQuotes Software Corp.

CodePage

Retourne la page de code de la langue du terminal client.

```
int  CodePage() const
```

Valeur de retour

La page de code de la langue du terminal client.

Note

Pour récupérer la page de code, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_CODEPAGE).

© 2000-2015, MetaQuotes Software Corp.

CPUCores

Retourne le nombre de coeurs CPU du système.

```
int  CPUCores() const
```

Valeur de retour

Le nombre de coeur CPU du système.

Note

Pour récupérer le nombre de coeurs CPU, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_CPU_CORES).

© 2000-2015, MetaQuotes Software Corp.

MemoryPhysical

Retourne la quantité de mémoire physique (en Mo).

```
int  MemoryPhysical() const
```

Valeur de retour

La quantité de mémoire physique (en Mo).

Note

Pour récupérer la mémoire physique, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_MEMORY_PHYSICAL).

© 2000-2015, MetaQuotes Software Corp.

MemoryTotal

Retourne la quantité de mémoire totale, disponible pour le terminal client ou l'agent (en Mo).

```
int  MemoryTotal() const
```

Valeur de retour

La quantité de mémoire totale (en Mo) disponible pour le terminal ou le client.

Note

Pour récupérer la mémoire totale, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_MEMORY_TOTAL).

© 2000-2015, MetaQuotes Software Corp.

MemoryAvailable

Retourne la quantité de mémoire libre, disponible pour pour le terminal client ou l'agent (en Mo).

```
int  MemoryTotal() const
```

Valeur de retour

La quantité de mémoire libre (en Mo), disponible pour pour le terminal client ou l'agent.

Note

Pour récupérer la mémoire libre, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_MEMORY_TOTAL).

© 2000-2015, MetaQuotes Software Corp.

MemoryUsed

Retourne la quantité de mémoire utilisée par le terminal client ou l'agent (en Mo).

```
int  MemoryUsed() const
```

Valeur de retour

La quantité de mémoire utilisée par le terminal ou l'agent (en Mo).

Note

Pour récupérer la quantité de mémoire, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_MEMORY_USED).

© 2000-2015, MetaQuotes Software Corp.

IsX64

Retourne le type du terminal client (32/64 bits).

```
bool  IsX64() const
```

Valeur de retour

vrai si la version 64 bits est utilisée, faux sinon.

Note

Pour récupérer le type du terminal client, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_X64).

© 2000-2015, MetaQuotes Software Corp.

OpenCLSupport

Retourne la version de OpenCL supportée par la carte vidéo.

```
int  OpenCLSupport() const
```

Valeur de retour

La valeur retournée est de la forme : 0x00010002 = "1,2". 0 signifie que OpenCL n'est pas supporté.

Note

Pour récupérer la version d'OpenCL, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_OPENCL_SUPPORT).

© 2000-2015, MetaQuotes Software Corp.

DiskSpace

Retourne la quantité d'espace disque disponible pour le terminal ou l'agent (en Mo).

```
int  MDiskSpace() const
```

Valeur de retour

Quantité d'espace disque disponible (en Mo), disponible pour le terminal client ou l'agent (pour les fichiers sauvegardés dans le répertoire MQL5\Files).

Note

Pour récupérer l'espace disque disponible, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_DISK_SPACE).

© 2000-2015, MetaQuotes Software Corp.

Language

Retourne la langue du terminal client.

```
string  Language() const
```

Valeur de retour

La langue utilisée dans le terminal client.

Note

Pour récupérer la langue, la fonction TerminalInfoString() est appelée (propriété TERMINAL_LANGUAGE).

© 2000-2015, MetaQuotes Software Corp.

Name

Retourne le nom du terminal client.

```
string  Name() const
```

Valeur de retour

Le nom du terminal client.

Note

Pour récupérer le nom du terminal client, la fonction TerminalInfoInteger() est appelée (propriété TERMINAL_NAME).

© 2000-2015, MetaQuotes Software Corp.

Company

Retourne le nom du courtier.

```
string  Company() const
```

Valeur de retour

Le nom du courtier.

Note

Pour récupérer le nom du courtier, la fonction TerminalInfoString() est appelée (propriété TERMINAL_COMPANY).

© 2000-2015, MetaQuotes Software Corp.

Path

Retourne le répertoire du terminial client.

```
string  Path() const
```

Valeur de retour

Le répertoire du terminal client.

Note

Pour récupérer le répertoire du terminal client, la fonction TerminalInfoString() est appelée (propriété TERMINAL_PATH).

© 2000-2015, MetaQuotes Software Corp.

DataPath

Retourne le répertoire de données (data folder) du terminal.

```
string  DataPath() const
```

Valeur de retour

Le répertoire des données du terminal client.

Note

Pour récupérer le répertoire des données du terminal, la fonction TerminalInfoString() est appelée (propriété TERMINAL_DATA_PATH).

© 2000-2015, MetaQuotes Software Corp.

CommonDataPath

Retourne le répertoire des données communes de tous les terminaux client installés sur l'ordinateur.

```
string  CommonDataPath() const
```

Valeur de retour

Le répertoire des données communes.

Note

Pour récupérer le répertoire des données communes, la fonction TerminalInfoString() est appelée (propriété COMMON_DATA_PATH).

© 2000-2015, MetaQuotes Software Corp.

InfoInteger

Retourne la valeur de la propriété correspondante de l'environnement du programme MQL5.

```
int  TerminalInfoInteger(
   int  property_id      // identifiant d'une propriété
   );
```

Paramètres

property_id

[in] Identifiant d'une propriété. Peut être une des valeurs de l'énumération ENUM_TERMINAL_INFO_INTEGER.

Valeur de retour

La valeur de type int.

Note

Ppur récupérer la valeur de la propriété, la fonction TerminalInfoInteger() est appelée.

© 2000-2015, MetaQuotes Software Corp.

InfoString

Retourne la valeur de la propriété correspondante de l'environnement du programme MQL5. La propriété doit être de type string.

```
string  TerminalInfoString(
   int  property_id      // identifiant d'une propriété
   );
```

Paramètres

property_id

[in] Identifiant d'une propriété. Peut être une des valeurs de l'énumération ENUM_TERMINAL_INFO_STRING.

Valeur de retour

La valeur de type string.

Note

Ppur récupérer la valeur de la propriété, la fonction TerminalInfoString() est appelée.

© 2000-2015, MetaQuotes Software Corp.

Classes de Stratégies de Trading

Cette section contient les détails techniques d'utilisation des classes de création et de test des stratégies de trading et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création de vos stratégies de trading.

La Bibliothèque Standard MQL5 (en terme de stratégies de trading) est située dans le répertoire de travail du terminal dans le répertoire Include\Expert.

Classes de base	Description
CExpertBase	Classe de base pour toutes les classes de stratégies de trading
CExpert	Classe de base pour les Expert Advisor
CExpertSignal	Classe de base pour les classes de Signaux de Trading
CExpertTrailing	Classe de base pour les classes de Trailing Stop (stops suiveurs)
CExpertMoney	Classe de base pour les classes de Money Management

Classes de signal de trading	Description
CSignalAC	Le module des signaux basés sur les modèles de marché de l'indicateur Accelerator Oscillator.
CSignalAMA	Le module des signaux basés sur les modèles de marché de l'indicateur Adaptive Moving Average.
CSignalAO	Le module des signaux basés sur les modèles de marché de l'indicateur Awesome Oscillator.
CSignalBearsPower	Le module des signaux basés sur les modèles de marché de l'oscillateur Bears Power.
CSignalBullsPower	Le module des signaux basés sur les modèles de marché de l'oscillateur Bulls Power.
CSignalCCI	Le module des signaux basés sur les modèles de marché de l'oscillateur Commodity Channel Index.
CSignalDeM	Le module des signaux basés sur les modèles de marché de l'oscillateur DeMarker.
CSignalDEMA	Le module des signaux basés sur les modèles de marché de l'indicateur Double Exponential Moving Average.

© 2000-2015, MetaQuotes Software Corp.

Classes de signal de trading	Description
CSignalEnvelopes	Le module des signaux basés sur les modèles de marché de l'indicateur Envelopes.
CSignalFrAMA	Le module des signaux basés sur les modèles de marché de l'indicateur Fractal Adaptive Moving Average.
CSignalITF	Le module de filtration des signaux par date/heure.
CSignalMACD	Le module des signaux basés sur les modèles de marché de l'oscillateur MACD.
CSignalMA	Le module des signaux basés sur les modèles de marché de l'indicateur Moving Average.
CSignalSAR	Le module des signaux basés sur les modèles de marché de l'indicateur Parabolic SAR.
CSignalRSI	Le module des signaux basés sur les modèles de marché de l'oscillateur Relative Strength Index.
CSignalRVI	Le module des signaux basés sur les modèles de marché de l'oscillateur Relative Vigor Index.
CSignalStoch	Le module des signaux basés sur les modèles de marché de l'oscillateur Stochastic.
CSignalTRIX	Le module des signaux basés sur les modèles de marché de l'oscillateur Triple Exponential Average.
CSignalTEMA	Le module des signaux basés sur les modèles de marché de l'indicateur Triple Exponential Moving Average.
CSignalWPR	Le module des signaux basés sur les modèles de marché de l'oscillateur Williams Percent Range.

Classes de Trailing Stop (Stops Suiveurs)	Description
CTrailingFixedPips	Cette classe implémente l'algorithme du Stop Suiveur basé sur des points fixes
CTrailingMA	Cette classe implémente l'algorithme du Stop Suiveur basé sur les valeurs d'une Moyenne Mobile
CTrailingNone	Classe de type stub, elle n'utilise aucun algorithme de Stop Suiveur
CTrailingPSAR	Cette classe implémente l'algorithme du Stop Suiveur basé sur les valeurs de l'indicateur Parabolic SAR

© 2000-2015, MetaQuotes Software Corp.

Classes de Money Management	Description
CMoneyFixedLot	Classe utilisant un algorithme basé sur le trading avec une taille de lot fixe.
CMoneyFixedMargin	Classe utilisant un algorithme basé sur le trading avec une marge fixe prédéfinie.
CMoneyFixedRisk	Classe utilisant un algorithme basé sur le trading avec un risque prédéfini.
CMoneyNone	Classe utilisant un algorithme basé sur le trading avec une taille de lot minimum permis.
CMoneySizeOptimized	Classe utilisant un algorithme basé sur le trading avec une taille de lot variable, suivant les résultats des deals précédents.

© 2000-2015, MetaQuotes Software Corp.

Classes de base des Expert Advisors

Cette section contient les détails techniques d'utilisation des classes de création et de test des stratégies de trading et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création de vos stratégies de trading.

La Bibliothèque Standard MQL5 (en terme de stratégies de trading) est située dans le répertoire de travail du terminal dans le répertoire Include\Expert.

Classe	Description
CExpertBase	Classe de base pour toutes les classes de stratégies de trading
CExpert	Classe de base pour les Expert Advisor
CExpertSignal	Classe de base pour les classes de Signaux de Trading
CExpertTrailing	Classe de base pour les classes de Trailing Stop (stops suiveurs)
CExpertMoney	Classe de base pour les classes de Money Management

© 2000-2015, MetaQuotes Software Corp.

CExpertBase

CExpertBase est la classe de base de la classe CExpert et de toutes les classes de stratégie de trading.

Description

CExpertBase fournit les données et méthodes qui sont communes à tous les objets de l'Expert Advisor.

Déclaration

```
class CExpertBase : public CObject
```

Titre

```
#include <Expert\ExpertBase.mqh>
```

Méthodes de Classe

Méthodes Publiques :

Initialisation	
virtual Init	Méthode d'initialisation de l'instance de classe
virtual ValidationSettings	Vérifie les paramètres
Paramètres	
Symbol	Définit le symbole
Period	Définit la période
Magic	Définit l'identifiant de l'Expert Advisor
Indicateurs et Séries de Données	
virtual SetPriceSeries	sDéfinit les pointeurs vers les séries de données externes (séries de prix)
virtual SetOtherSeries	Définit les pointeurs vers les séries de données externes (séries de données autres que les prix)
virtual InitIndicators	Initialise les indicators et les séries de données
Accés aux Données Protégées	
InitPhase	Retourne la phase actuelle de l'initialisation de l'objet
TrendType	Définit le type de tendance
UsedSeries	Retourne le masque de bits des séries de données utilisées
EveryTick	Définit le flag "Chaque tick"
Accés au Séries de Données	

© 2000-2015, MetaQuotes Software Corp.

Open	Retourne l'élément de la série Open (prix d'ouverture) par son index
High	Retourne l'élément de la série High (plus haut) par son index
Low	Retourne l'élément de la série Low (plus bas) par son index
Close	Retourne l'élément de la série Close (prix de clôture) par son index
Spread	Retourne l'élément de la série Spread par son index
Time	Retourne l'élément de la série Time par son index
TickVolume	Retourne l'élément de la série TickVolume par son index
RealVolume	Retourne l'élément de la série RealVolume (volume réel) par son index

Méthodes Protégées :

Initialisation des Séries de Données	
InitOpen	Méthode d'initialisation de la série Open (ouverture)
InitHigh	Méthode d'initialisation de la série High (plus haut)
InitLow	Méthode d'initialisation de la série Low (plus bas)
InitClose	Méthode d'initialisation de la série Close (prix de clôture)
InitSpread	Méthode d'initialisation de la série Spread
InitTime	Méthode d'initialisation de la série Time
InitTickVolume	Méthode d'initialisation de la série TickVolume
InitRealVolume	Méthode d'initialisation de la série RealVolume
Méthodes Services	
virtual PriceLevelUnit	Retourne l'unité du niveau de prix
virtual StartIndex	Retourne l'index de la première barre à analyser
virtual CompareMagic	Compare l'identifiant de l'Expert Advisor avec la valeur spécifiée.

© 2000-2015, MetaQuotes Software Corp.

InitPhase

Retourne la phase actuelle de l'initialisation de l'objet.

```
ENUM_INIT_PHASE  InitPhase()
```

Valeur de retour

Phase courant de l'initialisation de l'objet.

Note

L'initialisation de l'objet est faite de plusieurs phases :

1. Démarrage de l'initialisation.

- lancement - après le constructeur
- fin - après la fin avec succès de la méthode Init(...).
- permise - appel de la méthode Init(...)
- non permise - appel de la méthode ValidationSettings() et autres méthodes d'initialisation

2. Phase d'initialisation des paramètres. Durant cette phase, vous devez définir tous les paramètres des objets utilisés pour la création d'indicateurs.

- lancement - après la fin avec succès de la méthode Init(...)
- fin - après la fin avec succès de la méthode ValidationSettings()
- permise - appel des méthodes Symbol(...) et Period(...)
- non permise - appel des méthodes Init(...), SetPriceSeries(...), SetOtherSeries(...) et InitIndicators(...)

3. Vérification des paramètres.

- lancement - après la fin avec succès de la méthode ValidationSettings()
- fin - après la fin avec succès de la méthode InitIndicators(...)
- permise - appel des méthodes Symbol(...), Period(...) et InitIndicators(...)
- non permise - appel de toute autre méthode d'initialisation

4. Fin de l'initialisation.

- lancement - après la fin avec succès de la méthode InitIndicators(...)
- non permise - appel des méthodes d'initialisation

© 2000-2015, MetaQuotes Software Corp.

TrendType

Définit le type de tendance.

```
void  TrendType(
   M_TYPE_TREND    value          // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du type de tendance.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

UsedSeries

Retourne le masque de bits des séries de données utilisées.

```
int  UsedSeries()
```

Valeur de retour

La liste des séries de données utilisées sous forme d'un masque de bits.

Note

Si le bit est défini, la série de données correspondante est utilisée, s'il n'est pas défini, la série de données n'est pas utilisée.

Correspondance des bits des séries de données :

bit 0 - série de données Open,
bit 1 - série de données High,
bit 2 - série de données Low,
bit 3 - série de données Close,
bit 4 - série de données Spread,
bit 5 - série de données Time,
bit 6 - série de données TickVolume,
bit 7 - série de données RealVolume.

© 2000-2015, MetaQuotes Software Corp.

EveryTick

Définit le flag "Chaque tick".

```
void  EveryTick(
   bool    value         // flag
   )
```

Paramètres

value

[in] Nouvelle valeur du flag.

Valeur de retour

Aucune.

Note

Si le flag n'est pas défini, la méthode de traitement n'est appelée que sur la première barre de la période et du symbole courants.

© 2000-2015, MetaQuotes Software Corp.

Open

Retourne l'élément de la série Open (prix d'ouverture) par son index.

```
double  Open(
   int     ind          // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série Open situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2015, MetaQuotes Software Corp.

High

Retourne l'élément de la série High par son index.

```
double  High(
   int     ind          // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série High situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2015, MetaQuotes Software Corp.

Low

Retourne l'élément de la série Low (plus bas) par son index.

```
double  Low(
   int     ind          // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série Low situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2015, MetaQuotes Software Corp.

Close

Retourne l'élément de la série Close par son index.

```
double  Close(
   int    ind          // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série Close situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2015, MetaQuotes Software Corp.

Spread

Retourne l'élément de la série Spread par son index.

```
double  Spread(
   int    ind          // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série Spread situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2015, MetaQuotes Software Corp.

Time

Retourne l'élément de la série Time par son index

```
datetime  Time(
   int    ind          // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série Time situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2015, MetaQuotes Software Corp.

TickVolume

Retourne l'élément de la série TickVolume par son index.

```
long  TickVolume(
   int     ind          // Index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série TickVolume situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2015, MetaQuotes Software Corp.

RealVolume

Retourne l'élément de la série RealVolume (volume réel) par son index.

```
long  RealVolume(
   int    ind         // index
   )
```

Paramètres

ind

[in] Index de l'élément.

Valeur de retour

Retourne la valeur numérique de l'élément de la série de données RealVolume situé à l'index spécifié en cas de succès, sinon retourne EMPTY_VALUE.

Note

EMPTY_VALUE est retourné dans deux cas :

1. La série de données n'est pas utilisée (le bit correspondant n'est pas défini).
2. L'index de l'élément est en dehors de l'intervalle de la série (out of range).

© 2000-2015, MetaQuotes Software Corp.

Init

Initialise l'objet.

```
bool  Init(
   CSymbolInfo       symbol,    // symbole
   ENUM_TIMEFRAMES   period,    // période
   double            point      // point
   )
```

Paramètres

symbol

[in] Pointeur sur l'objet de type CSymbolInfo pour accéder aux informations du symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

point

[in] Le "poids" du point de 2/4 chiffres.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Symbol

Définit le symbole.

```
bool  Symbol(
   string    name     // symbole
   )
```

Paramètres

name

[in] Symbole.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La définition du symbole est nécessaire si l'objet utilise un symbole différent de celui défini au moment de l'initialisation.

© 2000-2015, MetaQuotes Software Corp.

Period

Définit la période.

```
bool  Period(
   ENUM_TIMEFRAMES  value     // période
   )
```

Paramètres

value

[in] Période.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La définition de la période est nécessaire si l'objet utilise une période différente de la période définie au moment de l'initialisation.

© 2000-2015, MetaQuotes Software Corp.

Magic

Définit l'identifiant de l'Expert Advisor.

```
void  Magic(
   ulong  value     // magic
   )
```

Paramètres

value

[in] Identifiant de l'Expert Advisor.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

SetPriceSeries

Définit les pointeurs vers les séries de données externes (séries de données autres que les prix).

```
virtual bool  SetPriceSeries(
   CiOpen*    open,         // pointeur
   CiHigh*    high,         // pointeur
   CiLow*     low,          // pointeur
   CiClose*   close         // pointeur
   )
```

Paramètres

open

[in] Pointeur sur la série de données Open (prix d'ouverture).

high

[in] Pointeur sur la série de données High (plus haut).

low

[in] Pointeur sur la série de données Low (plus bas).

close

[in] Pointeur sur la série de données Close (prix de clôture).

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La définition des pointeurs sur les séries de données externes (séries de données des prix) est nécessaire si l'objet utilise des séries de données d'un symbole ou d'une période différents de ceux définis au moment de l'initialisation.

© 2000-2015, MetaQuotes Software Corp.

SetOtherSeries

Définit les pointeurs vers les séries de données externes (séries de données autres que les prix).

```
virtual bool  SetOtherSeries(
   CiSpread*       spread,        // pointeur
   CiTime*         time,          // pointeur
   CiTickVolume*   tick_volume,   // pointeur
   CiRealVolume*   real_volume    // pointeur
   )
```

Paramètres

spread

[in] Pointeur sur la série de données Spread.

time

[in] Pointeur sur la série de données Time.

tick_volume

[in] Pointeur sur la série de données TickVolume.

real_volume

[in] Pointeur sur la série de données RealVolume.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La définition des pointeurs sur les séries de données externes (séries de données autres que les prix) est nécessaire si l'objet utilise des séries de données d'un symbole ou d'une période différents de ceux définis au moment de l'initialisation.

© 2000-2015, MetaQuotes Software Corp.

InitIndicators

Initialise tous les indicateurs et les séries de données.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Les séries de données ne sont initialisées que si l'objet utilise le symbole ou la période définit lors de l'initialisation et que le symbole ou la période sont différents.

© 2000-2015, MetaQuotes Software Corp.

InitOpen

Initialise la série de données Open (prix d'ouverture).

```
bool  InitOpen(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données Open n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2015, MetaQuotes Software Corp.

InitHigh

Initialise la série de données High (plus haut).

```
bool  InitHigh(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données High n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2015, MetaQuotes Software Corp.

InitLow

Initialise la série de données Low (plus bas).

```
bool  InitLow(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données Low n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2015, MetaQuotes Software Corp.

InitClose

Initialise la série de données Close (prix de clôture).

```
bool  InitClose(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données Clôture n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2015, MetaQuotes Software Corp.

InitSpread

Initialise la série de données Spread.

```
bool  InitSpread(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données Spread n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2015, MetaQuotes Software Corp.

InitTime

Initialise la série de données Time (date/heure).

```
bool  InitTime(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données Time n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2015, MetaQuotes Software Corp.

InitTickVolume

Initialise la série de données TickVolume.

```
bool  InitTickVolume(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données TickVolume n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2015, MetaQuotes Software Corp.

InitRealVolume

Initialise la série de données RealVolume (volume réel).

```
bool  InitRealVolume(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La série de données RealVolume n'est initialisée que si l'Expert Advisor utilise le symbole/la période, si différents du symbole/de la période définis à l'initialisation (et que la série de données est utilisée par la suite).

© 2000-2015, MetaQuotes Software Corp.

PriceLevelUnit

Retourne l'unité du niveau de prix.

```
virtual double  PriceLevelUnit()
```

Valeur de retour

Valeur de l'unité du niveau de prix.

Note

La méthode de la classe de base retourne le "poids" du point à 2/4 décimales.

© 2000-2015, MetaQuotes Software Corp.

StartIndex

Retourne l'index de la première barre à analyser.

```
virtual int  StartIndex()
```

Valeur de retour

L'index de la première barre à analyser.

Note

La méthode retourne 0 si le flag pour analyser la barre courante est défini à vrai (analyse depuis la barre courante). Si le flag n'est pas défini, retourne 1 (analyse depuis la dernière barre complétée).

© 2000-2015, MetaQuotes Software Corp.

CompareMagic

Compare l'identifiant de l'Expert Advisor avec la valeur spécifiée.

```
virtual bool  CompareMagic(
   ulong  magic    // Valeur de comparaison
   )
```

Paramètres

magic

[in] Valeur de comparaison.

Valeur de retour

vrai s'ils sont égaux, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CExpert

Classe de base pour toutes les classes de stratégies de trading. Des algorithmes sont intégrés permettant de travailler avec les séries de données et les indicateurs, ainsi qu'un ensemble de méthodes virtuelles pour les stratégies de trading.

Comment l'utiliser :

1. Préparez l'algorithme de la stratégie ;
2. Creéez votre propre classe, dérivant de la classe CExpert ;
3. Surchargez les méthodes virtuelles de votre classe avec vos propres algorithmes.

Description

La classe CExpert est un ensemble de méthodes virtuelles pour l'implémentation des stratégies de trading.

Déclaration

```
class CExpert : public CExpertBase
```

Titre

```
#include <Expert\Expert.mqh>
```

Méthodes de Classe

Initialisation	
Init	Méthode d'initialisation de l'instance de classe
virtual InitSignal	Initialise l'objet Signal de Trading
virtual InitTrailing	Initialise l'objet Stop Suiveur (Trailing Stop)
virtual InitMoney	Initialise l'objet Money Management
virtual InitTrade	Initialise l'objet Trade
virtual ValidationSettings	Vérifie les paramètres
virtual InitIndicators	Initialise les indicateurs et les séries de données
virtual InitParameters	Méthode d'initialisation des paramètres
virtual Deinit	Méthode de dé-initialisation de l'instance de classe
virtual DeinitSignal	Dé-initialise l'objet Signal de Trading
virtual DeinitTrailing	Dé-initialise l'objet Stop Suiveur (Trailing Stop)
virtual DeinitMoney	Dé-initialise l'objet Money Management
virtual DeinitTrade	Dé-initialise l'objet Trade

© 2000-2015, MetaQuotes Software Corp.

virtual DeinitIndicators	Dé-initialise les indicateurs et les séries de données
Paramètres	
Magic	Définit l'identifiant de l'Expert Advisor
MaxOrders	Retourne/Définit le nombre maximum d'ordres autorisés
OnTickProcess	Définit un flag pour traiter l'évènement "OnTick"
OnTradeProcess	Définit un flag pour traiter l'évènement "OnTrade"
OnTimerProcess	Définit un flag pour traiter l'évènement "OnTimer"
OnChartEventProcess	Définit un flag pour traiter l'évènement "OnChartEvent"
OnBookEventProcess	Définit un flag pour traiter l'évènement "OnBookEvent"
Méthodes de Traitement des Evènements	
OnTick	Gestionnaire d'évènement OnTick
OnTrade	Gestionnaire d'évènement OnTrade
OnTimer	Gestionnaire d'évènement OnTimer
OnChartEvent	Gestionnaire d'évènement OnChartEvent
OnBookEvent	Gestionnaire d'évènement OnBookEvent
Méthodes de Mise à Jour	
Refresh	Met à jour toutes les données
Traitement	
Processing	Algorithme principal de traitement
Méthodes d'Entrée sur le Marché	
CheckOpen	Vérifie les conditions d'ouverture d'une position
CheckOpenLong	Vérifie les conditions d'ouverture d'une position longue
CheckOpenShort	Vérifie les conditions d'ouverture d'une position courte (short)
OpenLong	Ouvre une position longue
OpenShort	Ouvre une position courte (short)
Méthodes de Sortie du Marché	
CheckClose	Vérifie les conditions pour clôturer la position

© 2000-2015, MetaQuotes Software Corp.

	courante
CheckCloseLong	Vérifie les conditions pour clôturer une position longue
CheckCloseShort	Vérifie les conditions pour clôturer une position courte (short)
CloseAll	Ferme toutes les positions ouvertes et supprime tous les ordres
Close	Ferme la position ouverte
CloseLong	Ferme la position longue
CloseShort	Ferme la position courte (short)
Méthodes de Renversement de Position	
CheckReverse	Vérifie les conditions pour renverser une position ouverte
CheckReverseLong	Vérifie les conditions pour renverser une position longue
CheckReverseShort	Vérifie les conditions pour renverser une position courte (short)
ReverseLong	Effectue le renversement d'une position longue
ReverseShort	Effectue le renversement d'une position courte (short)
Méthodes de Suivi d'une Position/ d'un Ordre	
CheckTrailingStop	Vérifie les conditions pour modifier les paramètres d'une position
CheckTrailingStopLong	Vérifie les conditions du Stop Suiveur (Trailing Stop) d'une position longue
CheckTrailingStopShort	Vérifie les conditions du Stop Suiveur (Trailing Stop) d'une position courte (short)
TrailingStopLong	Positionne le Stop Suiveur (Trailing Stop) d'une position longue
TrailingStopShort	Positionne le Stop Suiveur (Trailing Stop) d'une position courte (short)
CheckTrailingOrderLong	Vérifie les conditions du Stop Suiveur (Trailing Stop) d'un ordre en attente de type Buy Limit/ Stop
CheckTrailingOrderShort	Vérifie les conditions du Stop Suiveur (Trailing Stop) d'un ordre en attente de type Sell Limit/ Stop
TrailingOrderLong	Positionne le Stop Suiveur (Trailing Stop) d'un ordre en attente de type Buy Limit/Stop

© 2000-2015, MetaQuotes Software Corp.

TrailingOrderShort	Positionne le Stop Suiveur (Trailing Stop) d'un ordre en attente de type Sell Limit/Stop
Méthodes de Suppression d'un Ordre	
CheckDeleteOrderLong	Vérifie les conditions pour supprimer un ordre en attente de type Buy
CheckDeleteOrderShort	Vérifie les conditions pour supprimer un ordre en attente de type Sell
DeleteOrders	Supprime tous les ordres
DeleteOrder	Supprime un ordre en attente de type Stop/Limit
DeleteOrderLong	Supprime un ordre en attente de type Buy Limit/Stop
DeleteOrderShort	Supprime un ordre en attente de type Sell Limit/Stop
Méthodes liées aux Volumes de Trading	
LotOpenLong	Retourne le volume de trading pour une opération de type buy
LotOpenShort	Retourne le volume de trading pour une opération de type sell
LotReverse	Retourne le volume de trading pour une opération de renversement d'une position
Méthodes d'Historique de Trading	
PrepareHistoryDate	Définit la date de début de suivi de l'historique de trading
HistoryPoint	Crée un point de contrôle de l'historique de trading (sauvegarde le nombre de positions, les ordres, les deals et les ordres historiques)
CheckTradeState	Compare l'état courant avec celui sauvegardé et appelle le gestionnaire d'évènement correspondant
Flags d'Evènement	
WaitEvent	Définit le flag d'attente d'évènement
NoWaitEvent	Réinitialise le flag d'attente d'évènement
Méthodes de Traitement des Evènements de Trading	
TradeEventPositionStopTake	Gestionnaire de l'évènement "Stop Loss/Take Profit d'une position déclenché"
TradeEventOrderTriggered	Gestionnaire de l'évènement "Ordre en Attente Déclenché"

© 2000-2015, MetaQuotes Software Corp.

TradeEventPositionOpened	Gestionnaire de l'évènement "Position Ouverte"
TradeEventPositionVolumeChanged	Gestionnaire de l'évènement "Volume de la Position Modifié"
TradeEventPositionModified	Gestionnaire de l'évènement "Position Modifiée"
TradeEventPositionClosed	Gestionnaire de l'évènement "Position Fermée"
TradeEventOrderPlaced	Gestionnaire de l'évènement "Ordre en Attente Placé"
TradeEventOrderModified	Gestionnaire de l'évènement "Ordre en Attente Modifié"
TradeEventOrderDeleted	Gestionnaire de l'évènement "Ordre en Attente Supprimé"
TradeEventNotIdentified	Gestionnaire d'un évènement non identifié
Méthodes de Service	
TimeframeAdd	Ajoute une période à suivre
TimeframesFlags	Retourne le flag indiquant la période avec une nouvelle barre

© 2000-2015, MetaQuotes Software Corp.

Init

Méthode d'initialisation d'une instance de classe.

```
bool  Init(
   string              symbol,        // symbole
   ENUM_TIMEFRAMES     period,        // période
   bool                every_tick,    // flag
   ulong               magic          // magic
   )
```

Paramètres

symbol

[in] Symbole.

period

[in] Période (énumération ENUM_TIMEFRAMES).

every_tick

[in] Flag.

magic

[in] Identifiant de l'Expert Advisor (numéro Magic).

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si every_tick est à true, la méthode Processing() est appelée à chaque tick du symbole en cours. Sinon, la méthode Processing() n'est appelée qu'à l'apparition d'une nouvelle barre du symbole en cours.

© 2000-2015, MetaQuotes Software Corp.

Magic

Définit l'identifiant (magic) de l'Expert Advisor.

```
void  Magic(
   ulong  value     // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de l'identifiant de l'Expert Advisor.

Valeur de retour

Aucune.

Note

Cette fonction définit la valeur de l'identifiant (magic) de l'Expert Advisor aux classes suivantes : Trade, Signal, Money, Trailing.

Implémentation

```
//+------------------------------------------------------------------+
//| Définit le nombre magic à l'objet et ses dépendances. |
//| ENTREE :  value - nouvelle valeur du nombre magic. |
//| SORTIE : aucune.                                                 |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
void CExpert::Magic(ulong value)
  {
   if(m_trade!=NULL)     m_trade.SetExpertMagicNumber(value);
   if(m_signal!=NULL)    m_signal.Magic(value);
   if(m_money!=NULL)     m_money.Magic(value);
   if(m_trailing!=NULL) m_trailing.Magic(value);
//---
   CExpertBase::Magic(value);
  }
```

© 2000-2015, MetaQuotes Software Corp.

InitSignal

Initialise l'objet Signal de Trading.

```
virtual bool  InitSignal(
   CExpertSignal*     signal=NULL,        // pointeur
   )
```

Paramètres

signal

[in] Pointeur sur l'objet de classe CExpertSignal (ou une classe dérivée).

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si le paramètre signal est NULL, la classe CExpertSignal sera utilisée, elle ne fait rien.

© 2000-2015, MetaQuotes Software Corp.

InitTrailing

Initialise l'objet Stop Suiveur (Trailing Stop).

```
virtual bool  InitTrailing(
   CExpertTrailing*     trailing=NULL,        // pointeur
   )
```

Paramètres

trailing

[in] Pointeur sur un objet de classe CExpertTrailing (ou une classe dérivée).

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si le paramètre trailing est NULL, la classe ExpertTrailing sera utilisée, elle ne fait rien.

© 2000-2015, MetaQuotes Software Corp.

InitMoney

Initialise l'objet Money Management.

```
virtual bool  InitMoney(
   CExpertMoney*     money=NULL,         // pointeur
   )
```

Paramètres

money

[in] Pointeur sur un objet de classe CExpertMoney (ou un de ses descendants).

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si money est NULL, la classe CExpertMoney sera utilisée, elle utilise le lot minimum.

© 2000-2015, MetaQuotes Software Corp.

InitTrade

Initialise l'objet Trade.

```
virtual bool  InitTrade(
   ulong            magic,       // magic
   CExpertTrade*    trade=NULL   // pointeur
   )
```

Paramètres

magic

[in] Identifiant de l'Expert Advisor (sera utilisé dans les demandes de deals).

trade

[in] Pointeur sur un objet CExpertTrade.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Deinit

Méthode de dé-initialisation de l'instance de classe.

```
virtual void Deinit()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

OnTickProcess

Définit un flag pour traiter l'évènement OnTick.

```
void  OnTickOProcess(
   bool     value        // flag
   )
```

Paramètres

value

[in] Flag pour traiter l'évènement OnTick .

Valeur de retour

Aucune.

Note

Si le flag est à true, l'évènement OnTick est traité, par défaut, le flag est à true.

© 2000-2015, MetaQuotes Software Corp.

OnTradeProcess

Définit un flag pour traiter l'évènement OnTrade .

```
void  OnTradeProcess(
   bool     value        // flag
   )
```

Paramètres

value

[in] Flag pour traiter l'évènement OnTrade.

Valeur de retour

Aucune.

Note

Si le flag est à true, l'évènement OnTrade est traité, par défaut, le flag est à false.

© 2000-2015, MetaQuotes Software Corp.

OnTimerProcess

Définit un flag pour traiter l'évènement OnTimer.

```
void  OnTimerProcess(
   bool     value        // flag
   )
```

Paramètres

value

[in] Flag pour traiter l'évènement OnTimer.

Valeur de retour

Aucune.

Note

Si le flag est à true, l'évènement OnTimer est traité, par défaut, le flag est à false.

© 2000-2015, MetaQuotes Software Corp.

OnChartEventProcess

Définit un flag pour traiter l'évènement OnChartEvent.

```
void  OnChartEventProcess(
   bool     value        // flag
   )
```

Paramètres

value

[in] Flag pour traiter l'évènement OnChartEvent.

Valeur de retour

Aucune.

Note

Si le flag est à true, l'évènement OnChartEvent est traité, par défaut, le flag est à false.

© 2000-2015, MetaQuotes Software Corp.

OnBookEventProcess

Définit un flag pour traiter l'évènement Définit un flag pour traiter l'évènement OnBookEvent.

```
void  OnChartEventProcess(
   bool     value        // flag
   )
```

Paramètres

value

[in] Flag pour traiter l'évènement OnBookEvent .

Valeur de retour

Aucune.

Note

Si le flag est à true, l'évènement OnBookEvent est traité, par défaut, le flag est à false.

© 2000-2015, MetaQuotes Software Corp.

MaxOrders (Méthode "Get")

Retourne le nombre maximum d'ordres autorisés.

```
int  MaxOrders()
```

Valeur de retour

Nombre maximum d'ordres autorisés.

MaxOrders (Méthode "Set")

Définit le nombre maximum d'ordres autorisés.

```
void  MaxOrders(
   int      max_orders        // Nouvelle valeur
   )
```

Paramètres

max_orders

[in] Nouveau nombre maximum d'ordres autorisés.

Valeur de retour

Aucune.

Note

Par défaut, le nombre maximum d'ordres autorisés est 1.

© 2000-2015, MetaQuotes Software Corp.

Signal

Retourne le pointeur sur l'objet Signal de Trading.

```
CExpertSignal*  Signal() const
```

Valeur de retour

Pointeur sur l'objet Signal de Trading.

© 2000-2015, MetaQuotes Software Corp.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Vérifie aussi les paramètres de tous les objets de l'Expert Advisor.

© 2000-2015, MetaQuotes Software Corp.

InitIndicators

Initialise tous les indicateurs et les séries de données.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Les séries de données sont initialisées si l'objet utilise le symbole ou la période définit lors de l'initialisation et que le symbole ou la période sont différents. La fonction appelle alors la méthode virtuelle InitIndicators() des objets signal de trading, stop suiveur et money management.

© 2000-2015, MetaQuotes Software Corp.

OnTick

Gestionnaire de l'évènement OnTick.

```
virtual void  OnTick()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

OnTrade

Gestionnaire de l'évènement OnTrade.

```
virtual void  OnTrade()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

OnTimer

Gestionnaire de l'évènement OnTimer.

```
virtual void  OnTimer()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

OnChartEvent

Gestionnaire de l'évènement OnChartEvent.

```
virtual void  OnChartEvent(
   const int     id,       // identifiant d'évènement
   const long&   lparam,   // paramètre de type long
   const double  dparam,   // paramètre de type double
   const string  sparam    // paramètre de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long.

dparam

[in] Paramètre d'évènement de type double.

sparam

[in] Paramètre d'évènement de type string.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

OnBookEvent

Gestionnaire de l'évènement OnBookEvent.

```
virtual void  OnBookEvent(
   const string&     symbol       // symbole
)
```

Paramètres

symbol

[in] Symbole de l'évènementOnBookEvent.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

InitParameters

Initialise les paramètres de l'Expert Advisor.

```
virtual bool  InitParameters()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La fonction InitParameters() de la classe de base CExpert n'effectue aucun traitement et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

DeinitTrade

Dé-initialise l'objet Trade.

```
virtual void  DeinitTrade()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

DeinitSignal

Dé-initialise l'objet Signal.

```
virtual void  DeinitSignal()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

DeinitTrailing

Dé-initialise l'objet Suiveur.

```
virtual void  DeinitTrailing()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

DeinitMoney

Dé-initialise l'objet Money Management.

```
virtual void  DeinitMoney()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

DeinitIndicators

Dé-initialise tous les indicateurs et les séries de données.

```
virtual void  DeinitIndicators()
```

Valeur de retour

Aucune.

Note

Cette fonction dé-initialise également tous les indicateurs et les séries de données de tous les objets auxiliaires.

© 2000-2015, MetaQuotes Software Corp.

Refresh

Met à jour toutes les données

```
virtual bool  Refresh()
```

Valeur de retour

vrai si le traitement d'un autre tick est recquis, faux sinon.

Note

Permet de déterminer le besoin de traitement d'un tick. Si besoin, cette fonction met à jour toutes les cotations, les séries de données et les données des indicateurs puis retourne vrai.

Implémentation

```
//+------------------------------------------------------------------+
//| Rafraîchissement des données à traiter                           |
//| ENTREE : aucune.                                                 |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::Refresh()
  {
   MqlDateTime time;
//--- rafraîchit les cotations
   if(!m_symbol.RefreshRates()) return(false);
//--- vérifie le besoin de traitement
   TimeToStruct(m_symbol.Time(),time);
   if(m_period_flags!=WRONG_VALUE && m_period_flags!=0)
      if((m_period_flags & TimeframesFlags(time))==0) return(false);
   m_last_tick_time=time;
//--- rafraîchit les indicateurs
   m_indicators.Refresh();
//--- ok
   return(true);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Processing

Algorithme principal de traitement.

```
virtual bool  Processing()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Effecute les étapes suivantes :

1. Vérifie l'existence d'une position ouverte sur le symbole. Si aucune position n'est trouvée, passe à l'étape №5.
2. Vérifie les conditions pour retourner une position ouverte (méthode CheckReverse()). Si la position a été retournée, la fonction se termine.
3. Vérifie les conditions pour fermer la position (méthode CheckClose()). Si la position a été clôturée, passe à l'étape №5.
4. Vérifie les conditions pour modifier les paramètres de la position (méthode CheckTrailingStop()). Si les paramètres de la position ont été modifiés, la fonction se termine.
5. Vérifie l'existence d'ordres en attente sur le symbole. Si aucun ordre en attente n'est trouvé, passe à l'étape №9.
6. Vérifie les conditions pour supprimer un ordre (méthode CheckDeleteOrderLong() pour les ordres en attente de type buy ou CheckDeleteOrderShort() pour les ordres en attente de type sell). Si l'ordre a été supprimé, passe à l'étape №9.
7. Vérifie les conditions pour modifier les paramètres d'un ordre en attente (CheckTrailingOrderLong() pour les ordres de type buy ou CheckTrailingOrderShort() pour les ordres de type sell). Si les paramètres de l'ordre ont été modifiés, la fonction se termine.
8. Sortie.
9. Vérifie les conditions pour ouvrir une position (méthode CheckOpen()).

Si vous voulez implémenter votre propre algorithem, vous devez surcharger la méthode Processing() de la classe dérivée.

Implémentation

© 2000-2015, MetaQuotes Software Corp.

```
//+------------------------------------------------------------------+
//| Fonction principale                                              |
//| ENTREE : aucune.                                                 |
//| SORTIE : vrai si une opération a été effectuée, faux sinon.      |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::Processing()
  {
//--- vérifie l'existence de positions ouvertes
   if(m_position.Select(m_symbol.Name()))
     {
      //--- ouvre la position
      //--- vérifie la possibilité de renverser la position
      if(CheckReverse()) return(true);
      //--- vérifie la possibilité de fermer la position/supprimer les ordres en atter
      if(!CheckClose())
        {
         //--- vérifie la possibilité de modifier la position
         if(CheckTrailingStop()) return(true);
         //--- retour sans traitement
         return(false);
        }
     }
//--- vérifie la présence d'ordres en attente
   int total=OrdersTotal();
   if(total!=0)
     {
      for(int i=total-1;i>=0;i--)
        {
         m_order.SelectByIndex(i);
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         if(m_order.OrderType()==ORDER_TYPE_BUY_LIMIT || m_order.OrderType()==ORDER_TY
           {
            //--- vérifie la possibilité de supprimer un ordre long en attente
            if(CheckDeleteOrderLong()) return(true);
            //--- vérifie la possibilité de modifier un ordre long en attente
            if(CheckTrailingOrderLong()) return(true);
           }
         else
           {
            //--- vérifie la possibilité de supprimer un ordre sell en attente
            if(CheckDeleteOrderShort()) return(true);
            //--- vérifie la possibilité de modifier un ordre sell en attente
            if(CheckTrailingOrderShort()) return(true);
           }
         //--- retour sans traitement
         return(false);
        }
     }
//--- vérifie la possibilité d'ouvrir une position / de mettre un ordre en attente
   if(CheckOpen()) return(true);
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckOpen

Vérifie les conditions pour ouvrir une position.

```
virtual bool  CheckOpen()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

Vérifie les conditions pour ouvrir des positions longues (CheckOpenLong()) et courtes ou short (CheckOpenShort()).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour ouvrir une position ou définir un ordre limit/stop |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckOpen()
  {
   if(CheckOpenLong())  return(true);
   if(CheckOpenShort()) return(true);
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckOpenLong

Vérifie les conditions pour ouvrir une position longue.

```
virtual bool  CheckOpenLong()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

La fonction vérifie les conditions pour ouvrir une position longue (méthode CheckOpenLong() de l'objet Signal) et ouvre la position longue (méthode OpenLong()) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour ouvrir une position longue / définir un ordre limit/stop |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- vérifie le signal pour ouvrir une opération longue
   if(m_signal.CheckOpenLong(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenLong(price,sl,tp));
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckOpenShort

Vérifie les conditions pour ouvrir une position courte.

```
virtual bool  CheckOpenShort()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

La fonction vérifie les conditions pour ouvrir une position courte ou short (méthode CheckOpenShort() de l'objet Signal) et ouvre cette position (méthode OpenShort()) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour ouvrir une position short ou définir / ordre limit/stop |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- vérifie le signal pour ouvrir une opération short
   if(m_signal.CheckOpenShort(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenShort(price,sl,tp));
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

OpenLong

Ouvre une position longue.

```
virtual bool  OpenLong(
   double    price,    // prix
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

tp

[in] Prix du Take Profit

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Retourne le volume de trading (méthode LotOpenLong(...)) et ouvre une position long (méthode Buy() de l'objet Trade) si le volume de trading n'est pas égal à 0.

Implémentation

```
//+------------------------------------------------------------------+
//| Ouvre une position long ou définit un ordre limit/stop.          |
//| ENTREE :  price - prix,                                          |
//|           sl    - stop loss,                                     |
//|           tp    - take profit.                                   |
//| SORTIE : vrai si opération réalisée avec succès, sinon faux. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::OpenLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- récupère le lot pour l'ouverture
   double lot=LotOpenLong(price,sl);
//--- vérifie le lot pour l'ouverture
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

© 2000-2015, MetaQuotes Software Corp.

OpenShort

Ouvre une position courte (short)

```
virtual bool  OpenShort(
   double    price,    // prix
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

tp

[in] Prix du Take Profit

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Retourne le volume de trading (méthode LotOpenShort(...)) et ouvre une position short (méthode Sell() de l'objet Trade) si le volume de trading n'est pas égal à 0.

Implémentation

```
//+------------------------------------------------------------------+
//| Ouvre une position short ou définit un ordre limit/stop.    |
//| ENTREE :  price - prix,                                          |
//|           sl    - stop loss,                                     |
//|           tp    - take profit.                                   |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon.     |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::OpenShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- retourne le lot pour l'ouverture
   double lot=LotOpenShort(price,sl);
//--- vérifie le lot pour l'ouverture
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckReverse

Vérifie les conditions pour renverser une position ouverte.

```
virtual bool  CheckReverse()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

Vérifie les conditions pour renverser une position longue (CheckReverseLong()) ou courte (CheckReverseShort()).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérification pour renverser une position                         |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckReverse()
  {
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- vérifie la possibilité de renverser une position longue
      if(CheckReverseLong())  return(true);
     }
   else
      //--- vérifie la possibilité de renverser une position courte
      if(CheckReverseShort()) return(true);
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckReverseLong

Vérifie les conditions pour renverser une position longue.

```
virtual bool  CheckReverseLong()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

La fonction vérifie les conditions pour renverser une position longue (méthode CheckReverseLong() de l'objet Signal) et effectue le renversement de la position longue actuelle (méthode ReverseLong(...)) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Vérification pour renverser une position longue                  |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- vérifie le signal de retournement d'une position longue
   if(m_signal.CheckReverseLong(price,sl,tp,expiration)) return(ReverseLong(price,sl,t
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckReverseShort

Vérifie les conditions pour renverser une position courte (short).

```
virtual bool  CheckReverseLong()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

La fonction vérifie les conditions pour renverser une position courte (ou short) (méthode CheckReverseShort() de l'objet Signal) et effectue le renversement de la position short actuelle (méthode ReverseShort(...)) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Vérification pour renverser une position courte                  |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- vérifie le signal de retournement d'une position short
   if(m_signal.CheckReverseShort(price,sl,tp,expiration)) return(ReverseShort(price,sl
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

ReverseLong

Effectue le renversement d'une position longue.

```
virtual bool  ReverseLong(
   double    price,    // prix
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

tp

[in] Prix du Take Profit

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Récupère le volume du renversement de la position (méthode LotReverse()) et effectue le retournement de la position longue (méthode Sell() de l'objet Trade) si le volume de trading n'est pas égal à 0.

Implémentation

```
//+------------------------------------------------------------------+
//| Renversement d'une position longue                               |
//| ENTREE :  price - prix,                                          |
//|           sl    - stop loss,                                     |
//|           tp    - take profit.                                   |
//| SORTIE : vrai si opération réalisée avec succès, sinon faux.     |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::ReverseLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- récupère le lot pour le renversement
   double lot=LotReverse(sl);
//--- vérifie le lot
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

© 2000-2015, MetaQuotes Software Corp.

ReverseShort

Effectue le renversement d'une position courte (short).

```
virtual bool  ReverseShort(
   double    price,    // prix
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

tp

[in] Prix du Take Profit

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Récupère le volume du renversement de la position (méthode LotReverse()) et effectue le retournement de la position courte (méthode Buy() de l'objet Trade) si le volume de trading n'est pas égal à 0.

Implémentation

```
//+------------------------------------------------------------------+
//| Renversement d'une position courte                               |
//| ENTREE :  price - prix,                                          |
//|           sl    - stop loss,                                     |
//|           tp    - take profit.                                   |
//| SORTIE : vrai si opération réalisée avec succès, sinon faux.     |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::ReverseShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- récupère le lot pour le renversement
   double lot=LotReverse(sl);
//--- vérifie le lot
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckClose

Vérifie les conditions de clôture d'une position.

```
virtual bool  CheckClose()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

1. Cette fonction vérifie les conditions de sortie (Stop Out) de l'Expert Advisor (méthode CheckClose() de l'objet money management). Si la condition est satisfaite, cette fonction ferme la position et supprime tous les ordres (CloseAll(...)) et sort.
2. Elle vérifie les conditions pour fermer une position longue ou courte (méthodes CheckCloseLong() ou CheckCloseShort()) et si la position est fermée, elle supprime tous les ordres (méthode DeleteOrders()).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour fermer une position ou supprimer un ordre limite/stop|
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckClose()
  {
   double lot;
//--- la position doit être sélectionnée avant l'appel
   if((lot=m_money.CheckClose(GetPointer(m_position)))!=0.0)
      return(CloseAll(lot));
//--- vérifie le type de position
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- vérifie la possibilité de fermer la position longue / supprimer les ordres
      if(CheckCloseLong())
        {
         DeleteOrders();
         return(true);
        }
     }
   else
     {
      //--- vérifie la possibilité de fermer la position courte / supprimer les ordres
      if(CheckCloseShort())
        {
         DeleteOrders();
         return(true);
        }
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckCloseLong

Vérifie les conditions pour clôturer une position longue.

```
virtual bool  CheckCloseLong()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction vérifie les conditions pour clôturer une position longue (méthode CheckCloseLong() de l'objet Signal). Si les conditions sont réunies, elle ferme la position ouverte (méthode CloseLong(...)).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour fermer une position longue ou supprimer un ordre limit/stop |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseLong()
  {
   double price=EMPTY_VALUE;
//--- vérifie les opérations pour clôturer les opérations sur les positions longues
   if(m_signal.CheckCloseLong(price))
      return(CloseLong(price));
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckCloseShort

Vérifie les conditions pour clôturer une position courte (short).

```
virtual bool  CheckCloseShort()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction vérifie les conditions pour clôturer une position courte (ou short) (méthode CheckCloseShort() de l'objet Signal). Si les conditions sont réunies, elle ferme la position ouverte (méthode CloseShort(...)).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour fermer une position courte ou supprimer un ordre limit/stop |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseShort()
  {
   double price=EMPTY_VALUE;
//--- vérifie les opérations pour clôturer les opérations sur les positions short
   if(m_signal.CheckCloseShort(price))
      return(CloseShort(price));
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CloseAll

Ferme une partie ou toutes les positions.

```
virtual bool  CloseAll(
   double    lot    // lot
   )
```

Paramètres

lot

[in] Nombre de lots pour réduire la position.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Cette fonction effectue une clôture partielle ou totale des positions (méthodes Sell() et Buy() de l'objet CTrade pour les positions long/short) et supprime tous les ordres (méthode DeleteOrders()).

Implémentation

```
//+------------------------------------------------------------------+
//| Clôture d'une position et suppression des ordres                 |
//| ENTREE :  lot - volume à clôturer.                               |
//| SORTIE : vrai si l'opération a été réalisée avec succès, sinon faux. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CloseAll(double lot)
  {
   bool result;
//--- vérifie les opérations
   if(m_position.PositionType()==POSITION_TYPE_BUY) result=m_trade.Sell(lot,0,0,0);
   else                                             result=m_trade.Buy(lot,0,0,0);
   result|=DeleteOrders();
//---
   return(result);
  }
```

© 2000-2015, MetaQuotes Software Corp.

Close

Ferme les positions ouvertes.

```
virtual bool  Close()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Ferme la position (méthode PositionClose() de l'objet CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Ferme la position |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::Close()
  {
   return(m_trade.PositionClose(m_symbol.Name()));
  }
```

© 2000-2015, MetaQuotes Software Corp.

CloseLong

Ferme la position longue.

```
virtual bool  CloseLong(
   double    price    // prix
   )
```

Paramètres

price

[in] Prix.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Ferme la position long (méthode Sell(...) de l'objet CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Ferme une position longue                                        |
//| ENTREE :  price - prix de clôture. |
//| SORTIE : vrai si l'opération a été réalisée avec succès, sinon faux. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CloseLong(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Sell(m_position.Volume(),price,0,0));
  }
```

© 2000-2015, MetaQuotes Software Corp.

CloseShort

Ferme la position courte (short)

```
virtual bool  CloseShort(
   double    price    // prix
   )
```

Paramètres

price

[in] Prix.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Ferme la position courte (short) (méthode Buy(...) de l'objet CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Ferme une position short                                         |
//| ENTREE :  price - prix de clôture. |
//| SORTIE : vrai si opération réalisée avec succès, sinon faux. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CloseShort(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Buy(m_position.Volume(),price,0,0));
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStop

Vérifie les conditions du Stop Suiveur (Trailing Stop) de la position ouverte.

```
virtual bool  CheckTrailingStop()
```

Valeur de retour

vrai une opération de trading a été executée, faux sinon.

Note

Vérifie les conditions du Stop Suiveur (Trailing Stop) de la position ouverte (CheckTrailingStopLong() ou CheckTrailingStopShort() pour les positions longues ou courtes).

Implémentation

```
//+------------------------------------------------------------------+
//| Check for trailing stop/profit position                          |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStop()
  {
//--- la position doit d'abord être sélectionnée
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- vérifie la possibilité de modifier une position long
      if(CheckTrailingStopLong()) return(true);
     }
   else
     {
      //--- vérifie la possibilité de modifier une position short
      if(CheckTrailingStopShort()) return(true);
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopLong

Cette fonction vérifie les conditions du Stop Suiveur (Trailing Stop) de la position long ouverte.

```
virtual bool  CheckTrailingStopLong()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Cette fonction vérifie les conditions du Stop Suiveur (Trailing Stop) de la position long ouverte (méthode CheckTrailingStopLong(...) de l'objet Expert Trailing). Si les conditions sont réunies, elle modifie les paramètres de la position (méthode TrailingStopLong(...)).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérifie le stop/profit suiveur d'une position longue |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopLong()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- vérifie les opérations des stops suiveur des opérations long
   if(m_trailing.CheckTrailingStopLong(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- traitement du stop suiveur de l'opération
      return(TrailingStopLong(sl,tp));
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopShort

Cette fonction vérifie les conditions du Stop Suiveur (Trailing Stop) de la position courte ouverte.

```
virtual bool  CheckTrailingStopShort()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Cette fonction vérifie les conditions du Stop Suiveur (Trailing Stop) de la position courte ouverte (méthode CheckTrailingStopShort(...) de l'objet Expert Trailing). Si les conditions sont réunies, elle modifie les paramètres de la position (méthode TrailingStopShort(...)).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérifie le stop/profit suiveur d'une position short |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopShort()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- vérifie les opérations des stops suiveur des opérations short
   if(m_trailing.CheckTrailingStopShort(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- traitement du stop suiveur de l'opération short
      return(TrailingStopShort(sl,tp));
     }
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

TrailingStopLong

Modifie les paramètres d'une position longue ouverte.

```
virtual bool  TrailingStopLong(
   double    sl,    // Stop Loss
   double    tp,    // Take Profit
   )
```

Paramètres

sl

[in] Prix du Stop Loss

tp

[in] Prix du Take Profit

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction modifie les paramètres de la position longue ouverte (méthode PositionModify(...) de la classe d'objet CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Stop/profit suiveur pour une position longue                     |
//| ENTREE :  sl - nouveau stop loss,                                |
//|           tp - nouveau take profit.                              |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon.     |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopLong(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000-2015, MetaQuotes Software Corp.

TrailingStopShort

Modifie les paramètres d'une position courte ouverte.

```
virtual bool  TrailingStopLong(
   double    sl,    // Stop Loss
   double    tp,    // Take Profit
   )
```

Paramètres

sl

[in] Prix du Stop Loss

tp

[in] Take Profit price.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction modifie les paramètres de la position courte ouverte (méthode PositionModify(...) de la classe d'objet CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Stop/profit suiveur pour une position short                      |
//| ENTREE :  sl - nouveau stop loss,                                |
//|           tp - nouveau take profit.                              |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon.     |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopShort(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingOrderLong

Vérifie les conditions du Stop Suiveur (Trailing Stop) d'un ordre en attente de type Buy Limit/Stop

```
virtual bool  CheckTrailingOrderLong()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction vérifie les conditions du Stop Suiveur (Trailing Stop) pour un ordre Buy Limit/Stop (méthode CheckTrailingOrderLong() de l'objet Signal de Trading). Si les conditions sont réunies, les paramètres de l'ordre sont modifiés (méthode TrailingOrderLong(...)) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Check for trailing long limit/stop order                         |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderLong()
  {
   double price;
//--- vérifie la possibilité de modifier un ordre long
   if(m_signal.CheckTrailingOrderLong(GetPointer(m_order),price))
      return(TrailingOrderLong(m_order.PriceOpen()-price));
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingOrderShort

Vérifie les conditions du Stop Suiveur (Trailing Stop) d'un ordre en attente de type Sell Limit/Stop

```
virtual bool  CheckTrailingOrderShort()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

La fonction vérifie les conditions du Stop Suiveur (Trailing Stop) pour un ordre Sell Limit/Stop (méthode CheckTrailingOrderShort() de l'objet Signal de Trading). Si les conditions sont réunies, les paramètres de l'ordre sont modifiés (méthode TrailingOrderShort(...)) si nécessaire.

Implémentation

```
//+------------------------------------------------------------------+
//| Check for trailing short limit/stop order                        |
//| ENTREE : aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                                |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderShort()
  {
   double price;
//--- vérifie la possibilité de modifier un ordre short
   if(m_signal.CheckTrailingOrderShort(GetPointer(m_order),price))
      return(TrailingOrderShort(m_order.PriceOpen()-price));
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

TrailingOrderLong

Modifie les paramètres d'un ordre en attente Buy Limit/Stop.

```
virtual bool  TrailingOrderLong(
   double    delta    // delta
   )
```

Paramètres

delta

[in] Prix delta.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Modifie les paramètres d'un ordre en attente Buy Limit/Stop (méthode OrderModify(...) de la classe CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Ordre long limit/stop en attente |
//| ENTREE :  delta - changement de prix.                            |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune.                              |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderLong(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- modification de l'ordre long
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

© 2000-2015, MetaQuotes Software Corp.

TrailingOrderShort

Modifie les paramètres d'un ordre en attente Sell Limit/Stop.

```
virtual bool  TrailingOrderShort(
   double    delta    // delta
   )
```

Paramètres

delta

[in] Prix delta.

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Modifie les paramètres d'un ordre en attente Sell Limit/Stop (méthode OrderModify(...) de la classe CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Ordre short limit/stop en attente                                |
//| ENTREE :  delta - changement de prix.                            |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune.                                |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderShort(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- modification de l'ordre short
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckDeleteOrderLong

Vérifie les conditions pour supprimer un ordre en attente de type Buy Limit/Stop.

```
virtual bool  CheckDeleteOrderLong()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Cette fonction vérifie l'heure d'expiration d'un ordre. La fonction vérifie les conditions pour supprimer un ordre Sell Limit/Stop (méthode CheckCloseLong() de l'objet Signal). Si les conditions sont réunies, elle supprime l'ordre (méthode DeleteOrderLong()).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour supprimer un ordre long limit/stop                   |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderLong()
  {
   double price;
//--- vérifie la possibilité de supprimer un ordre long
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderLong());
     }
   if(m_signal.CheckCloseLong(price))
      return(DeleteOrderLong());
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

CheckDeleteOrderShort

Vérifie les conditions pour supprimer un ordre en attente de type Sell Limit/Stop.

```
virtual bool  CheckDeleteOrderShort()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Cette fonction vérifie l'heure d'expiration d'un ordre. La fonction vérifie les conditions pour supprimer un ordre en attente Sell Limit/Stop (méthode CheckCloseShort() de l'objet Signal). Si les conditions sont réunies, l'ordre est supprimé (méthode DeleteOrderShort()).

Implémentation

```
//+------------------------------------------------------------------+
//| Vérif. pour supprimer un ordre short limit/stop |
//| ENTREE :  aucune. |
//| SORTIE : vrai si réalisé avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderShort()
  {
   double price;
//--- vérifie la possibilité de supprimer un ordre short
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderShort());
     }
   if(m_signal.CheckCloseShort(price))
      return(DeleteOrderShort());
//--- retour sans traitement
   return(false);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteOrders

Supprime tous les ordres.

```
virtual bool  DeleteOrders()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Supprime tous les ordres (DeleteOrder() pour tous les ordres).

Implémentation

```
//+------------------------------------------------------------------+
//| Supprime tous les ordres limite/stop                             |
//| ENTREE : aucune.                                                 |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrders()
  {
   bool result=false;
   int  total=OrdersTotal();
//---
   for(int i=total-1;i>=0;i--)
     {
      if(m_order.Select(OrderGetTicket(i)))
        {
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         result|=DeleteOrder();
        }
     }
//---
   return(result);
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteOrder

Supprime un ordre en attete Limit/Stop.

```
virtual bool  DeleteOrder()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Supprime l'ordre en attente Limit/Stop (méthode OrderDelete(...) de la classe CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Supprime un ordre limit/stop                |
//| ENTREE : aucune.                                                 |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune.                                       |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrder()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteOrderLong

Supprime un ordre en attente de type Buy Limit/Stop.

```
virtual bool  DeleteOrderLong()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Supprime l'ordre en attente Buy Limit/Stop (méthode OrderDelete(...) de la classe CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Supprime un ordre long limit/stop                                |
//| ENTREE : aucune.                                                 |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune.                                               |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderLong()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2015, MetaQuotes Software Corp.

DeleteOrderShort

Supprime un ordre en attente de type Sell Limit/Stop.

```
virtual bool  DeleteOrderShort()
```

Valeur de retour

vrai si l'opération a été éxécutée, sinon faux.

Note

Supprime l'ordre en attente Sell Limit/Stop (méthode OrderDelete(...) de la classe CTrade).

Implémentation

```
//+------------------------------------------------------------------+
//| Supprime l'ordre short limit/stop                                |
//| ENTREE : aucune.                                                 |
//| SORTIE : vrai si opération réalisée avec succès, faux sinon. |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderShort()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2015, MetaQuotes Software Corp.

LotOpenLong

Retourne le volume de trading pour une opération de type buy

```
double  LotOpenLong(
   double    price,    // prix
   double    sl        // Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading (en lots) pour une opération de type buy.

Note

Cette fonction retourne le volume de trading pour une opération de type buy (méthode CheckOpenLong(...) de l'objet money management).

Implémentation

```
//+------------------------------------------------------------------+
//| Méthode pour récupérer le nb de lots d'une position long ouverte. |
//| ENTREE :  price - prix,                                          |
//|           sl    - stop loss.                                     |
//| SORTIE : nb de lots.                                             |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
double CExpert::LotOpenLong(double price,double sl)
  {
   return(m_money.CheckOpenLong(price,sl));
  }
```

© 2000-2015, MetaQuotes Software Corp.

LotOpenShort

Retourne le volume de trading pour une opération de type sell.

```
double  LotOpenShort(
   double    price,   // prix
   double    sl       // Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss.

Valeur de retour

Volume de trading (en lots) pour une opération de type sell.

Note

Cette fonction retourne le volume de trading pour une opération de type sell (méthode CheckOpenShort(...) de l'objet money management).

Implémentation

```
//+------------------------------------------------------------------+
//| Méthode pour récupérer le nb de lots d'une position short ouverte.|
//| ENTREE :  price - prix,                                          |
//|           sl    - stop loss.                                     |
//| SORTIE : nb de lots.                                             |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
double CExpert::LotOpenShort(double price,double sl)
  {
   return(m_money.CheckOpenShort(price,sl));
  }
```

© 2000-2015, MetaQuotes Software Corp.

LotReverse

Retourne le volume de trading pour une opération de renversement d'une position.

```
double  LotReverse(
   double    sl       // Stop Loss
   )
```

Paramètres

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading (en lots) pour une opération de renversement de position.

Note

Cette fonction retourne le volume de trading pour une opération de renversement de position (méthode CheckReverse(...) de l'objet money management).

Implémentation

```
//+------------------------------------------------------------------+
//| Méthode pour récupérer le nb de lots pour un renversement de position. |
//| ENTREE :  sl - stop loss.                                        |
//| SORTIE : nb de lots.                                             |
//| REMARQUE : aucune. |
//+------------------------------------------------------------------+
double CExpert::LotReverse(double sl)
  {
   return(m_money.CheckReverse(GetPointer(m_position),sl));
  }
```

© 2000-2015, MetaQuotes Software Corp.

PrepareHistoryDate

Définit la date de début de suivi de l'historique de trading.

```
void  PrepareHistoryDate()
```

Note

La période de suivi de l'historique de trading est défini à partir du début du mois (mais pas moins de 1 jour).

© 2000-2015, MetaQuotes Software Corp.

HistoryPoint

Crée un point de contrôle de l'historique de trading (sauvegarde le nombre de positions, les ordres, les deals et les ordres historiques).

```
void  HistoryPoint(
   bool   from_check_trade=false   // flag
   )
```

Paramètres

from_check_trade=false

[in] Flag pour éviter la récursion.

Note

Sauvegarde le montant des positions, des ordres, des deals et l'historique des transactions.

© 2000-2015, MetaQuotes Software Corp.

CheckTradeState

Compare l'état courant avec celui sauvegardé et appelle le gestionnaire d'évènement correspondant.

```
bool  CheckTradeState()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

Elle vérifie le nombre de positions, d'ordres, de deals et les ordres historiques en comparant les valeurs sauvegardées par la méthode HistoryPoint() . Si l'historique de trading a changé, cette fonction appelle le gestionnaire d'évènement virtuel correspondant.

© 2000-2015, MetaQuotes Software Corp.

WaitEvent

Définit le flag d'attente d'évènement.

```
void  WaitEvent(
   ENUM_TRADE_EVENTS    event         // flag
   )
```

Paramètres

event

[in] Flag avec la liste d'évènements à définir (énumération ENUM_TRADE_EVENTS).

Valeur de retour

Aucune.

Flags d'Evènement

```
//--- flags des évènements attendus
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT              =0,         // aucun évènement attendu
   TRADE_EVENT_POSITION_OPEN         =0x1,       // en attente de l'évènement "ouvertu
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,       // en attente de l'évènement "modifi
   TRADE_EVENT_POSITION_MODIFY       =0x4,       // en attente de l'évènement "modifi
   TRADE_EVENT_POSITION_CLOSE        =0x8,       // en attente de l'évènement "clôture
   TRADE_EVENT_POSITION_STOP_TAKE    =0x10,      // en attente de l'évènement "déclenc
   TRADE_EVENT_ORDER_PLACE           =0x20,      // en attente de l'évènement "placeme
   TRADE_EVENT_ORDER_MODIFY          =0x40,      // en attente de l'évènement "modifi
   TRADE_EVENT_ORDER_DELETE          =0x80,      // en attente de l'évènement "suppres
   TRADE_EVENT_ORDER_TRIGGER         =0x100      // en attente de l'évènement "déclenc
  };
```

© 2000-2015, MetaQuotes Software Corp.

NoWaitEvent

Réinitialise le flag d'attente d'évènement.

```
void  NoWaitEvent(
   ENUM_TRADE_EVENTS    event         // flag
   )
```

Paramètres

event

[in] Flag avec la liste d'évènements à réinitialiser (énumération ENUM_TRADE_EVENTS).

Valeur de retour

Aucune.

Flags d'Evènement

```
//--- flags des évènements attendus
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT              =0,         // aucun évènement attendu
   TRADE_EVENT_POSITION_OPEN         =0x1,       // en attente de l'évènement "ouvertu
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,       // en attente de l'évènement "modific
   TRADE_EVENT_POSITION_MODIFY       =0x4,       // en attente de l'évènement "modific
   TRADE_EVENT_POSITION_CLOSE        =0x8,       // en attente de l'évènement "clôture
   TRADE_EVENT_POSITION_STOP_TAKE    =0x10,      // en attente de l'évènement "déclenc
   TRADE_EVENT_ORDER_PLACE           =0x20,      // en attente de l'évènement "placeme
   TRADE_EVENT_ORDER_MODIFY          =0x40,      // en attente de l'évènement "modific
   TRADE_EVENT_ORDER_DELETE          =0x80,      // en attente de l'évènement "suppres
   TRADE_EVENT_ORDER_TRIGGER         =0x100      // en attente de l'évènement "déclenc
  };
```

© 2000-2015, MetaQuotes Software Corp.

TradeEventPositionStopTake

Gestionnaire de l'évènement "Stop Loss/Take Profit d'une position déclenché".

```
virtual bool  TradeEventPositionStopTake()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

TradeEventOrderTriggered

Gestionnaire de l'évènement "Ordre en attente déclenché".

```
virtual bool  TradeEventOrderTriggered()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

TradeEventPositionOpened

Gestionnaire de l'évènement "Position ouverte".

```
virtual bool  TradeEventPositionOpened()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

TradeEventPositionVolumeChanged

Gestionnaire de l'évènement "Volume de position changé".

```
virtual bool  TradeEventPositionVolumeChanged()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

TradeEventPositionModified

Gestionnaire de l'évènement "Position modifiée".

```
virtual bool  TradeEventPositionModified()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

TradeEventPositionClosed

Gestionnaire de l'évènement "Position fermée".

```
virtual bool  TradeEventPositionClosed()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

TradeEventOrderPlaced

Gestionnaire de l'évènement "Ordre en attente placé".

```
virtual bool  TradeEventOrderPlaced()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

TradeEventOrderModified

Gestionnaire de l'évènement "Ordre en attente modifié".

```
virtual bool  TradeEventOrderModified()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

TradeEventOrderDeleted

Gestionnaire de l'évènement "Ordre enattente supprimé".

```
virtual bool  TradeEventOrderDeleted()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

TradeEventNotIdentified

Gestionnaire d'un évènement non identifié.

```
virtual bool  TradeEventNotIdentified()
```

Valeur de retour

La méthode de la classe CExpert ne fait rien et retourne toujours vrai.

Note

Différents évènements de trading peuvent arriver, dans ce cas, il est difficile de les identifier.

© 2000-2015, MetaQuotes Software Corp.

TimeframeAdd

Ajoute une période à surveiller.

```
void  TimeframeAdd(
   ENUM_TIMEFRAMES    period         // période
   )
```

Paramètres

period

[in] Période (énumération ENUM_TIMEFRAMES).

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

TimeframesFlags

La méthode retourne le flag indiquant les périodes d'une nouvelle barre.

```
int  TimeframesFlags(
   MqlDateTime&    time         // variable de date/heure
   )
```

Paramètres

time

[in] Variable de type MqlDateTime pour la nouvelle date/heure, passée par référence.

Valeur de retour

Retourne le flag indiquant les périodes d'une nouvelle barre.

© 2000-2015, MetaQuotes Software Corp.

CExpertSignal

CExpertSignal est la classe de base pour les signaux de trading, elle ne fait rien (exceptées les méthodes CheckReverseLong() et CheckReverseShort()) mais fournit l'interface.

Comment l'utiliser :

1. Préparez un algorithme pour les signaux de trading ;
2. Créez votre propre classe de signal de trading, dérivant de la classe CExpertSignal ;
3. Surchargez les méthodes virtuelles de votre classe avec vos propres algorithmes.

Vous pouvez trouver des exemples des classes de signaux de trading dans le répertoire Expert\Signal\.

Description

CExpertSignal est la classe de base pour l'implémentation des algorithmes des signaux de trading.

Déclaration

```
class CExpertSignal : public CExpertBase
```

Titre

```
#include <Expert\ExpertSignal.mqh>
```

Méthodes de Classe

Initialisation	
virtual InitIndicators	Initialise les indicateurs et les séries de données.
virtual ValidationSettings	Vérifie les paramètres
virtual AddFilter	Ajoute un filtre au signal combiné
Accés aux Données Protégées	
BasePrice	Définit le niveau du prix de base
UsedSeries	Retourne les flags des séries de données utilisées
Définition des Paramètres	
Weight	Définit la valeur du paramètres "Weight" (poids)
PatternsUsage	Définit la valeur du paramètre "PatternsUsage"
General	Définit la valeur du paramètre "General"
Ignore	Définit la valeur du paramètre "Ignore"
Invert	Définit la valeur du paramètre "Invert"
ThresholdOpen	Définit la valeur du paramètre "ThresholdOpen"

© 2000-2015, MetaQuotes Software Corp.

ThresholdClose	Définit la valeur du paramètre "ThresholdClose"
PriceLevel	Définit la valeur du paramètre "PriceLevel"
StopLevel	Définit la valeur du paramètre "StopLevel"
TakeLevel	Définit la valeur du paramètre "TakeLevel"
Expiration	Définit la valeur du paramètre "Expiration"
Magic	Définit la valeur du paramètre "Magic"
Vérifier les Conditions de Trading	
virtual CheckOpenLong	Vérifie les conditions d'ouverture d'une position longue
virtual CheckCloseLong	Vérifie les conditions pour clôturer une position longue
virtual CheckOpenShort	Vérifie les conditions d'ouverture d'une position courte (short)
virtual CheckCloseShort	Vérifie les conditions pour clôturer une position courte (short)
virtual CheckReverseLong	Vérifie les conditions pour renverser une position longue
virtual CheckReverseShort	Vérifie les conditions pour renverser une position courte
Définir les Paramètres de Trading	
virtual OpenLongParams	Définit les paramètres pour ouvrir une position longue
virtual OpenShortParams	Définit les paramètres pour ouvrir une position courte
virtual CloseLongParams	Définit les paramètres pour fermer une position longue
virtual CloseShortParams	Définit les paramètres pour fermer une position courte
Vérifier les Conditions des Ordres Suiveurs	
virtual CheckTrailingOrderLong	Vérifie les conditions pour modifier les paramètres d'un ordre d'achat (Buy) en attente
virtual CheckTrailingOrderShort	Vérifie les conditions pour modifier les paramètres d'un ordre de vente (Sell) en attente
Méthodes pour Vérifier les Ordres de Marché	
virtual LongCondition	Retourne le résultat de la vérification des conditions d'achat
virtual ShortCondition	Retourne le résultat de la vérification des

© 2000-2015, MetaQuotes Software Corp.

	conditions de vente
virtual Direction	Retourne la direction "pondérée" du prix

© 2000-2015, MetaQuotes Software Corp.

BasePrice

Définit le niveau du prix de base.

```
void  BasePrice(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du niveau du prix de base.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

UsedSeries

Retourne les flags des séries de données utilisées.

```
int  BasePrice()
```

Valeur de retour

Les flags des séries de données utilisées (si le symbole et la période correspondent au symbole et à la période en cours), sinon 0.

© 2000-2015, MetaQuotes Software Corp.

Weight

Définit la nouvelle valeur du paramètre "Weight".

```
void  Weight(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "Weight".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

PatternUsage

Définit la nouvelle valeur du paramètre "PatternsUsage".

```
void  PatternUsage(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "PatternsUsage".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

General

Définit la nouvelle valeur du paramètre "General".

```
void  General(
   int     value          // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "General".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Ignore

Définit la nouvelle valeur du paramètre "Ignore".

```
void  Ignore(
   long     value          // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "Ignore".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Invert

Définit la nouvelle valeur du paramètre "Invert".

```
void  Invert(
   long     value          // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "Invert".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

ThresholdOpen

Définit la nouvelle valeur du paramètre "ThresholdOpen".

```
void  ThresholdOpen(
   long     value          // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "ThresholdOpen".

Valeur de retour

Aucune.

Note

L'intervalle du paramètre "ThresholdOpen" est compris entre 0 et 100. Utilisé lors du "vote" pour ouvrir une position.

© 2000-2015, MetaQuotes Software Corp.

ThresholdClose

Définit la valeur du paramètre "ThresholdClose".

```
void  ThresholdOpen(
   long     value          // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "ThresholdClose".

Valeur de retour

Aucune.

Note

L'intervalle du paramètre "ThresholdClose" est compris entre 0 et 100. Utilisé lors du "vote" pour fermer une position.

© 2000-2015, MetaQuotes Software Corp.

PriceLevel

Définit la nouvelle valeur du paramètre "PriceLevel"

```
void  PriceLevel(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "PriceLevel".

Valeur de retour

Aucune.

Note

La valeur du paramètre "PriceLevel" est définie en unités de prix. La valeur numérique de l'unité de prix est retournée par la méthode PriceLevelUnit() . Le "PriceLevel" est utilisé pour définir le prix d'ouverture relativement au prix de base.

© 2000-2015, MetaQuotes Software Corp.

StopLevel

Définit la valeur du paramètre "StopLevel".

```
void  StopLevel(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "StopLevel".

Valeur de retour

Aucune.

Note

La valeur du paramètre "StopLevel" est définie en unités de prix. La valeur numérique de l'unité de prix est retournée par la méthode PriceLevelUnit() <t3>. Le "StopLevel" est utilisé pour définir le prix du Stop Loss relativement au prix d'ouverture.</t3>

© 2000-2015, MetaQuotes Software Corp.

TakeLevel

Définit la valeur du paramètre "TakeLevel"

```
void  TakeLevel(
   double    value         // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "TakeL".

Valeur de retour

Aucune.

Note

La valeur du paramètre "TakeLevel" est définie en unités de prix. La valeur numérique de l'unité de prix est retournée par la méthode PriceLevelUnit() <t3>. Le "TakeLevel" est utilisé pour définir le prix du Take Profit relativement au prix d'ouverture.</t3>

© 2000-2015, MetaQuotes Software Corp.

Expiration

Définit la valeur du paramètre "Expiration".

```
void  Expiration(
   int     value          // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "Expiration".

Valeur de retour

Aucune.

Note

La valeur du paramètre "Expiration" est définie en barres. Elle est utilisée comme date/heure d'expiration pour les ordres en attente (en cas de trading utilisant les ordres en attentes).

© 2000-2015, MetaQuotes Software Corp.

Magic

Définit la valeur du paramètre "Magic".

```
void  Magic(
   int    value          // Nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur du paramètre "Magic" (identifiant de l'Expert Advisor).

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

InitIndicators

Initialise tous les indicateurs et les séries de données.

```
virtual bool  InitIndicators(
   CIndicators*  indicators    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur sur une collection d'indicateurs et de séries de données.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Les séries de données ne sont initialisées que si l'objet utilise le symbole ou la période définit lors de l'initialisation et que le symbole ou la période sont différents.

© 2000-2015, MetaQuotes Software Corp.

AddFilter

Ajoute un filtre au signal composite.

```
virtual bool  InitIndicators(
   CExpertSignal*  filter    // pointeur
   )
```

Paramètres

indicators

[in] Pointeur vers un objet de filtre.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CheckOpenLong

Vérifie les conditions pour ouvrir une position longue.

```
virtual bool  CheckOpenLong(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CheckOpenShort

Vérifie les conditions pour ouvrir une position courte.

```
virtual bool  CheckOpenShort(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OpenLongParams

Définit les paramètres pour ouvrir une position longue.

```
virtual bool  OpenLongParams(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OpenShortParams

Définit les paramètres pour ouvrir une position courte.

```
virtual bool  OpenShortParams(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CheckCloseLong

Vérifie les conditions pour clôturer une position longue.

```
virtual bool  CheckCloseLong(
   double&  price      // prix
   )
```

Paramètres

price

[in][out] Variable pour le prix de clôture, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CheckCloseShort

Vérifie les conditions pour clôturer une position courte (short).

```
virtual bool  CheckCloseShort(
   double&  price      // prix
   )
```

Paramètres

price

[in][out] Variable pour le prix de clôture, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CloseLongParams

Définit les paramètres pour fermer une position longue.

```
virtual bool  CloseLongParams(
   double&    price          // prix
   )
```

Paramètres

price

[in][out] Variable pour le prix de clôture, passée par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CloseShortParams

Définit les paramètres pour fermer une position courte.

```
virtual bool  CloseShortParams(
   double&    price          // prix
   )
```

Paramètres

price

[in][out] Variable pour le prix de clôture, passée par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CheckReverseLong

Vérifie les conditions pour renverser une position longue.

```
virtual bool  CheckReverseLong(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CheckReverseShort

Vérifie les conditions pour renverser une position courte.

```
virtual bool  CheckReverseShort(
   double&    price,          // prix
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Paramètres

price

[in][out] Variable pour le renversement de prix, passée par référence.

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

expiration

[in][out] Variable pour la date/heure d'expiration, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingOrderLong

Vérifie les conditions pour modifier les paramètres d'un ordre d'achat (Buy) en attente.

```
virtual bool  CheckTrailingOrderLong(
   COrderInfo*     order,          // ordre
   double&         price           // prix
   )
```

Paramètres

order

[in] Pointeur vers un objet de classe COrderInfo.

price

[in][out] Variable du prix du Stop Loss

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingOrderShort

Vérifie les conditions pour modifier les paramètres d'un ordre de vente (Sell) en attente.

```
virtual bool  CheckTrailingOrderShort(
   COrderInfo*     order,          // ordre
   double&         price           // prix
   )
```

Paramètres

order

[in] Pointeur vers un objet de classe COrderInfo.

price

[in][out] Variable du prix du Stop Loss

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

LongCondition

Vérifie les conditions pour ouvrir une position longue.

```
virtual int  LongCondition()
```

Valeur de retour

Si les conditions sont réunies, retourne une valeur comprise entre 1 et 100 (dépendant de la "puissance" du signal), s'il n'y a aucun signal pour ouvrir une position longue, retourne 0.

Note

La méthode LongCondition() de la classe de base n'implémente pas la vérification des conditions pour ouvrir une position longue et retourne toujours 0.

© 2000-2015, MetaQuotes Software Corp.

ShortCondition

Vérifie les conditions pour ouvrir une position courte.

```
virtual int  ShortCondition()
```

Valeur de retour

Si les conditions sont réunies, retourne une valeur comprise entre 1 et 100 (dépendant de la "puissance" du signal), s'il n'y a aucun signal pour ouvrir une position courte, retourne 0.

Note

La méthode ShortCondition() de la classe de base n'implémente pas la vérification des conditions pour ouvrir une position courte et retourne toujours 0.

© 2000-2015, MetaQuotes Software Corp.

Direction

Retourne la valeur de la direction "pondérée".

```
virtual double  Direction()
```

Valeur de retour

Retourne une valeur supérieure à 0 si la direction est haussière (probablement) et inférieure à 0 si la direction est baissière. La valeur absolue dépend de la "puissance" du signal.

Note

Si les filtres sont utilisés, le résultat dépendra des filtres.

© 2000-2015, MetaQuotes Software Corp.

CExpertTrailing

CExpertTrailing est la classe de base pour les algorithmes des objets suiveurs, elle est une interface et n'effectue rien.

Comment l'utiliser :

1. Préparez un algorithme pour le suivi ;
2. Creéez votre propre classe, dérivant de la classe CExpertTrailing ;
3. Surchargez les méthodes virtuelles de votre classe avec vos propres algorithmes.

Vous pouvez trouver des exemples de classes de suivi dans le répertoire Expert\Trailing\.

Description

CExpertTrailing est la classe de base pour l'implémentation d'algorithmes de stops suiveurs.

Déclaration

```
class CExpertTrailing : public CExpertBase
```

Titre

```
#include <Expert\ExpertTrailing.mqh>
```

Méthodes de Classe

Vérifier les Conditions du Stop Suiveur	
virtual CheckTrailingStopLong	Vérifie les conditions pour modifier les paramètres de la position longue
virtual CheckTrailingStopShort	Vérifie les conditions pour modifier les paramètres de la position courte

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopLong

Vérifie les conditions pour modifier les paramètres de la position longue.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // pointeur
   double&         sl,           // Stop Loss
   double&         tp            // Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet de classe CPositionInfo<t3>.</t3>

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

La méthode CheckTrailingStopLong() de la classe de base retourne toujours faux.

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopShort

Vérifie les conditions pour modifier les paramètres de la position courte.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // pointeur
   double&         sl,           // Stop Loss
   double&         tp            // Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet de classe CPositionInfo<t3>.</t3>

sl

[in][out] Variable du prix du Stop Loss, passée par référence.

tp

[in][out] Variable du prix du Take Profit, passée par référence.

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

La méthode CheckTrailingStopShort() de la classe de base retourne toujours faux.

© 2000-2015, MetaQuotes Software Corp.

CExpertMoney

CExpertMoney est la classe de base pour les algorithmes de money et de risk management (gestion de votre capital et du risque).

Description

CExpertMoney est la classe de base pour implémenter les classes de money et de risk management.

Déclaration

```
class CExpertMoney : public CObject
```

Titre

```
#include <Expert\ExpertMoney.mqh>
```

Méthodes de Classe

Accés aux Données Protégées	
Percent	Définit la valeur du paramètre "Pourcentage de risque"
Initialisation	
virtual ValidationSettings	Vérifie les paramètres
Vérification des Conditions de Trading	
virtual CheckOpenLong	Retourne le volume d'une position longue
virtual CheckOpenShort	Retourne le volume d'une position courte
virtual CheckReverse	Retourne le volume pour renverser une position
virtual CheckClose	Vérifie les conditions pour clôturer une position ouverte

© 2000-2015, MetaQuotes Software Corp.

Percent

Définit la valeur du paramètre "Pourcentage de risque".

```
void  Percent(
   double  percent      // pourcentage de risque
   )
```

Paramètres

percent

[in] Pourcentage de risque.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La méthode ValidationSettings() de la classe de base retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

CheckOpenLong

Retourne le volume d'une position longue.

```
virtual double  CheckOpenLong(
   double  price,     // prix
   double  sl         // Stop Loss
   )
```

Paramètres

price

[in] Prix d'ouverture d'une position longue.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

© 2000-2015, MetaQuotes Software Corp.

CheckOpenShort

Retourne le volume d'une position courte.

```
virtual double  CheckOpenShort(
   double  price,     // prix
   double  sl         // Stop Loss
   )
```

Paramètres

price

[in] Prix d'ouverture d'une position courte.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position courte.

© 2000-2015, MetaQuotes Software Corp.

CheckReverse

Retourne le volume pour renverser une position.

```
virtual double  CheckReverse(
   CPositionInfo*  position,     // pointeur
   double          sl            // Stop Loss
   )
```

Paramètres

position

[in] Pointeur sur un objet de classe CPositionInfo.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume pour renverser la position.

© 2000-2015, MetaQuotes Software Corp.

CheckClose

Vérifie les conditions pour clôturer une position ouverte.

```
virtual double  CheckClose()
```

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

Modules des Signaux de Trading

Le terminal client inclut en standard un ensemble de modules de signaux de trading prêts-à-l'emploi pour l'"Assistant MQL5". Lors de la création d'un Expert Advisor avec l'Assistant MQL5, vous pouvez utilisez n'importe quelle combinaison des modules des signaux de trading (jusqu'à 64). La décision finale d'une opération de trading est faite sur la base d'une analyse complexe des singaux obtenus de tous les modules inclus. La description détaillée du mécanisme de la prise de décision est donnée ci-dessous.

Les modules de signaux suivants sont disponibles en standard :

- Signaux de l'indicateur Oscillateur d'Accélération
- Signaux de l'Indicateur Moyenne Mobile Adaptative
- Signaux de l'indicateur Awesome Oscillator
- Signaux de l'indicateur Oscillateur Bears Power
- Signaux de l'indicateur Oscillateur Bulls Power
- Signaux de l'indicateur Oscillateur Commodity Channel Index
- Signaux de l'indicateur Oscillateur DeMarker
- Signaux de l'indicateur Double Exponential Moving Average
- Signaux de l'indicateur Envelopes
- Signaux de l'indicateur Fractal Adaptive Moving Average
- Signaux du Filtre Horaire Intraday
- Signaux de l'oscillateur MACD
- Signaux de l'indicateur Moving Average
- Signaux de l'indicateur Parabolic SAR
- Signaux de l'oscillateur Relative Strength Index
- Signaux de l'oscillateur Relative Vigor Index
- Signaux de l'oscillateur Stochastic
- Signaux de l'oscillateur Triple Exponential Average
- Signaux de l'indicateur Triple Exponential Moving Average
- Signaux de l'oscillatuer Williams Percent Range

Mécanisme de Prise de Décision sur la Base des Modules de Signaux

Le mécanisme de prise de décision peut être représenté comme la liste suivante des principes de base :

- Chaque module de signal a ses propres modules de marché (une combinaison de prix et de valeurs d'un indicateur).
- Chaque modèle de marché a une valeur qui peut varier dans l'intervalle 1 à 100. Plus cette valeur est haute, plus le modèle est fort.
- Chaque modèle génère une prévision de la direction du mouvement des prix.
- La prévision d'un module est le résultat de la recherche de modèles intégrés, et est représentée comme un chiffre dans l'intervalle -100 à 100. Le signe détermine la direction de la prévision du mouvement (un signe négatif signifie que les prix vont baisser, un chiffre positif que les prix vont

© 2000-2015, MetaQuotes Software Corp.

monter). La valeur absolue correspond à la puissance du meilleur modèle trouvé.

- La prévision de chaque module est envoyé pour un "vote" final avec un coefficient de 0 à 1 spécifié dans ses paramètres (paramètre "Weight").
- Le résultat du vote est un nombre dans l'intervalle -100 à 100, où le signe détermine la direction de la prévision du mouvement, et la valeur absolue caractérise la puissance du signal. Il est calculé comme étant la moyenne arithmétique des prévisions pondétées de tous les modules de signaux.

Chaque Expert Advisor généré a deux paramètres modifiables — les niveaux de déclenchement d'ouverture et de clôture d'une position (ThresholdOpen et ThresholdClose) qui peuvent être égaux à une valeur de l'intervalle 0 à 100. Si la puissance du signal final est supérieur au niveau de déclenchement, une opération de trading correspondant au signe du signal est effectuée.

Exemples

Considérons un Expert Advisor avec les niveaux de déclenchement suivants : ThresholdOpen=20 et ThresholdClose=90. Deux modules de signaux participent à la prise de décision des opérations de trading : le module MA avec un poids de 0,4 et le module Stochastic avec un poids de 0,8. Analysons deux possibilités de signaux de trading obtenus :

Possibilité 1.

Le prix a croisé la Moyenne Mobile (MA) haussière à la hausse. Ce cas correspond à un des modèles de marché implémenté dans le module MA. Ce modèle implique une hausse des prix. Sa valeur est égale à 100. En même temps, l'oscillateur Stochastic s'est retourné à la baisse et forme une divergence avec les prix. Ce cas correspond à un des modèle de marché implémenté dans le module Stochastic. Ce modèle implique une baisse des prix. Le poids de ce modèle est 80.

Calculons le résult final du "vote". La valeur obtenue du module MA est 0,4 * 100 = 40. La valeur du module Stochastic est 0,8 * (-80) = -64. La valeur finale est la moyenne arithmétique de ces 2 valeurs : (40 - 64)/2 = -12. Le résultat du vote est un signal de vente avec une force relative de 12. Le niveau de déclenchement qui est à 20 n'est pas atteint. Aucune opération de trading ne sera donc déclenchée.

Possibilité 2.

Le prix a croisé la Moyenne Mobile (MA) haussière à la baisse. Ce cas correspond à un des modèles implémenté dans le module MA. Ce modèle implique une hausse des prix. Sa valeur est égale à 10. En même temps, l'oscillateur Stochastic s'est retourné à la baisse et forme une divergence avec les prix. Ce cas correspond à un des modèle de marché implémenté dans le module Stochastic. Ce modèle implique une baisse des prix. Le poids de ce modèle est 80.

Calculons le résult final du "vote". La valeur obtenue du module MA est 0,4 * 10 = 4. La valeur du module Stochastic est 0,8 * (-80) = -64. La valeur finale est la moyenne arithmétique de ces 2 valeurs : (4 - 64)/2 = -30. Le résultat du vote est un signal de vente avec une force relative de 30. Le niveau de déclenchement qui est à 20 est atteint. Le résultat est l'obtention d'un signal d'ouverture d'une position courte.

© 2000-2015, MetaQuotes Software Corp.

a) Divergence des prix avec l'oscillateur Stochastic (possibilités 1 et 2).

b) Les prix ont croisé la Moyenne Mobile à la hausse (possibilité 1).

c) Les prix ont croisé la Moyenne Mobile à la baisse (possibilité 2).

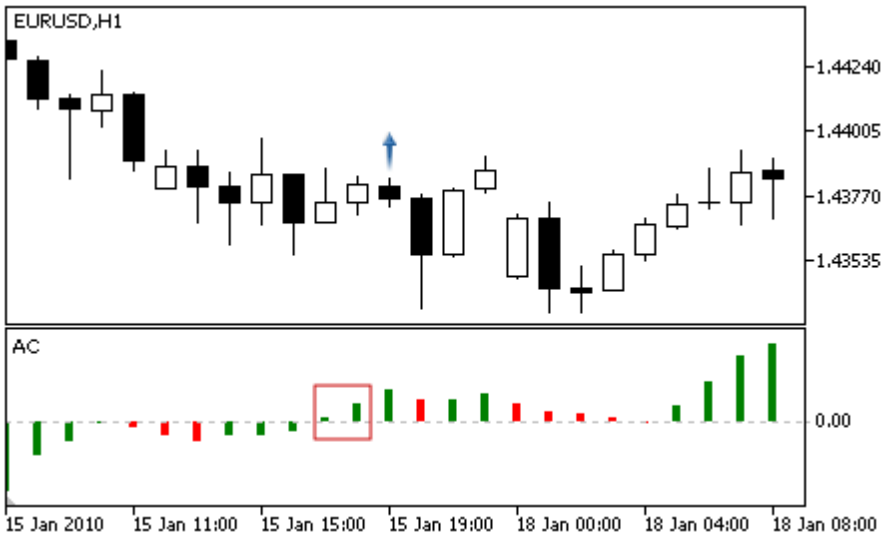
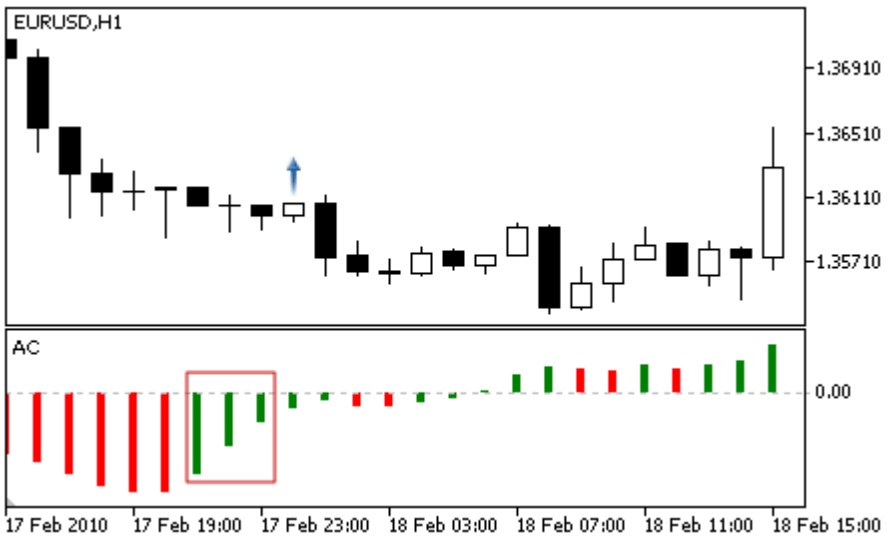
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Oscillateur d'Accélération

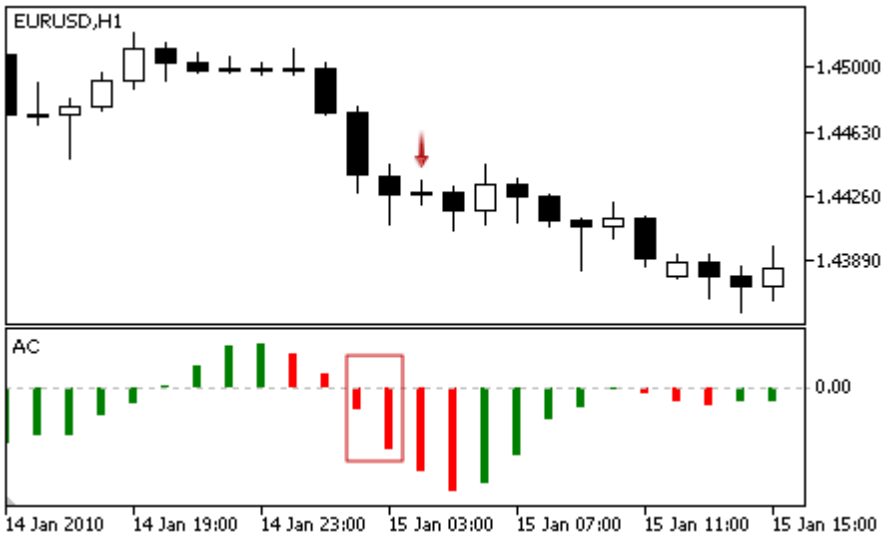
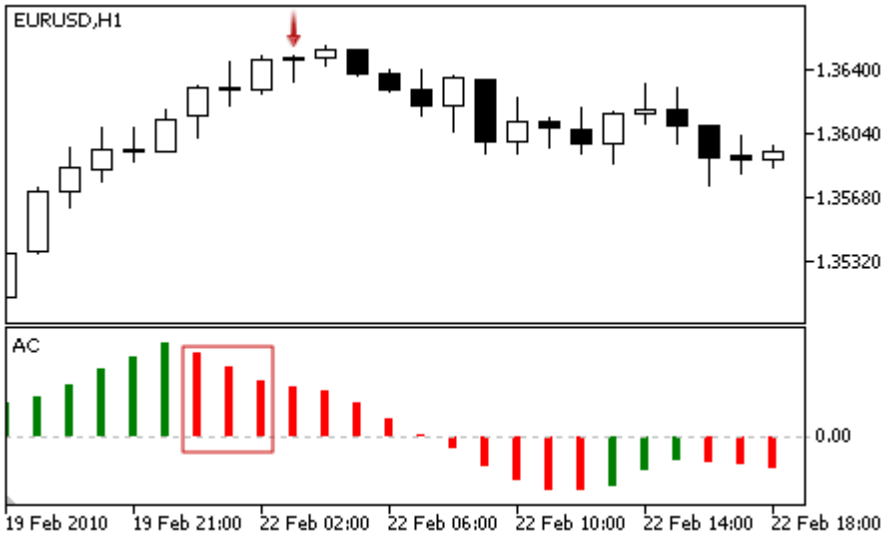
Ce module est basé sur les modèles de marché de l'indicateur Oscillateur d'Accélération. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• La valeur de l'indicateur est supérieure à 0 et augmente sur la barre analysée et la barre précédente.  • La valeur de l'indicateur est inférieure à 0 et augmente sur la barre analysée et la barre précédente. 
For selling	• La valeur de l'indicateur est inférieure à 0 et diminue sur la barre analysée et la barre précédente.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	 • La valeur de l'indicateur est inférieure à 0 et diminue sur la barre analysée et la barre précédente. 
Aucune objection pour l'achat	La valeur de l'indicateur augmente sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

© 2000-2015, MetaQuotes Software Corp.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.

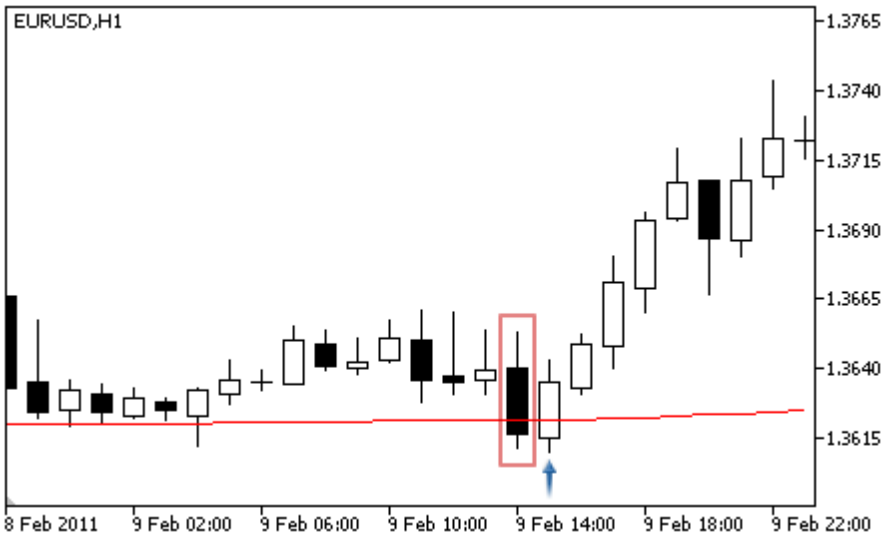
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'Indicateur Moyenne Mobile Adaptative

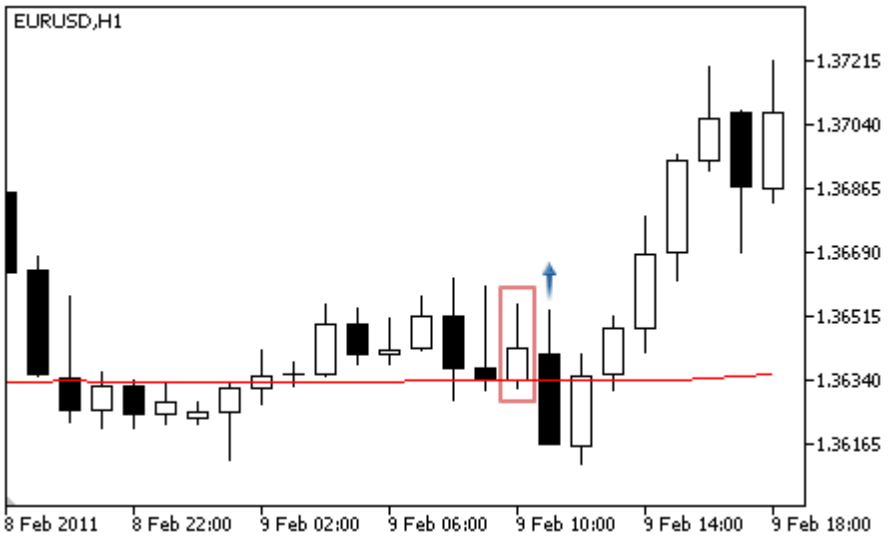
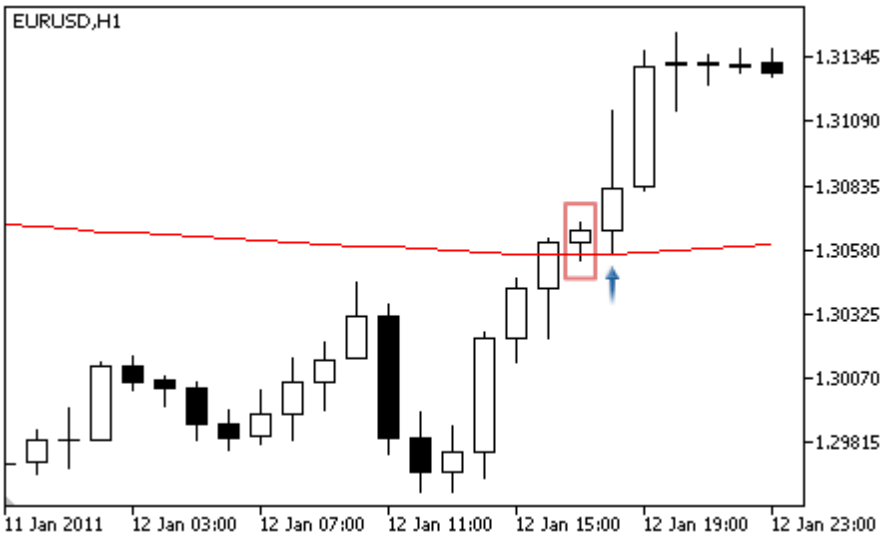
Ce module est basé sur les modèles de marché de l'indicateur Adaptive Moving Average (Moyenne Mobile Adaptative). Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

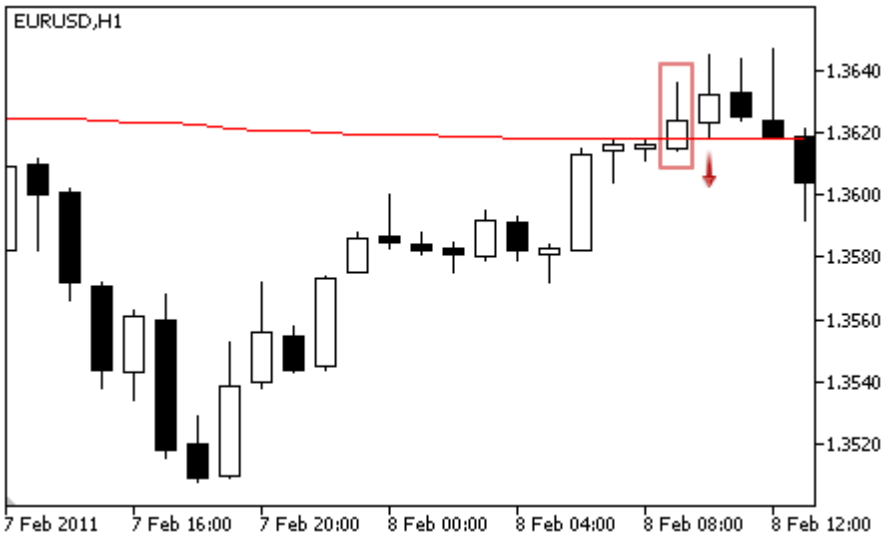
Type de Signal	Description des Conditions
Pour l'achat	• Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en hausse (signal fort).

© 2000-2015, MetaQuotes Software Corp.

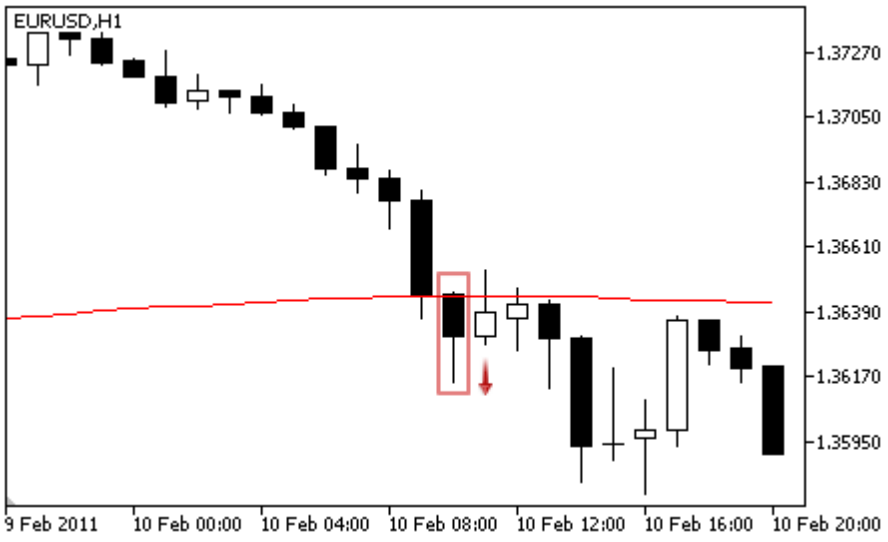
Type de Signal	Description des Conditions
	 • L'ombre inférieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont au-dessus de l'indicateur, et le prix le plus bas est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible). 
For selling	• Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions

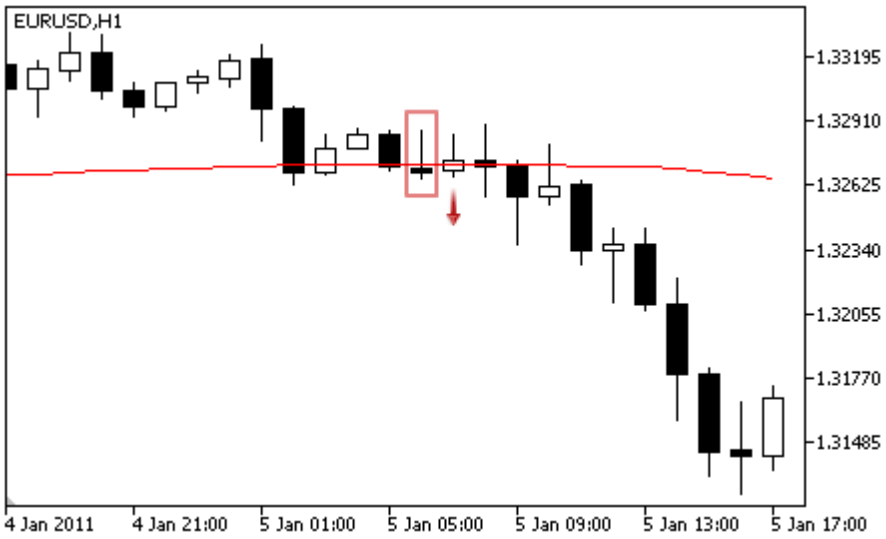


- **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en baisse (signal fort).



- L'ombre supérieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont en-dessous de l'indicateur, et le prix le plus haut est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	
Aucune objection pour l'achat	Les prix sont au-dessus de l'indicateur.
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de lissage de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

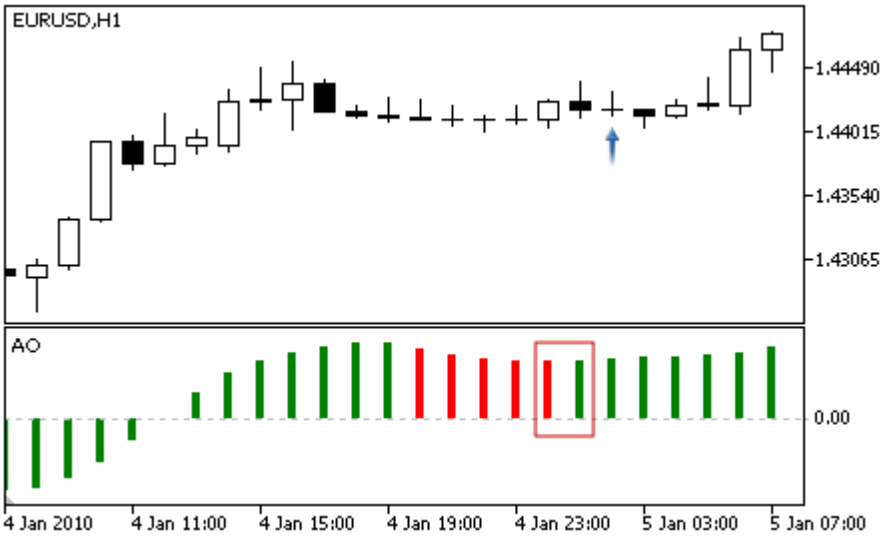
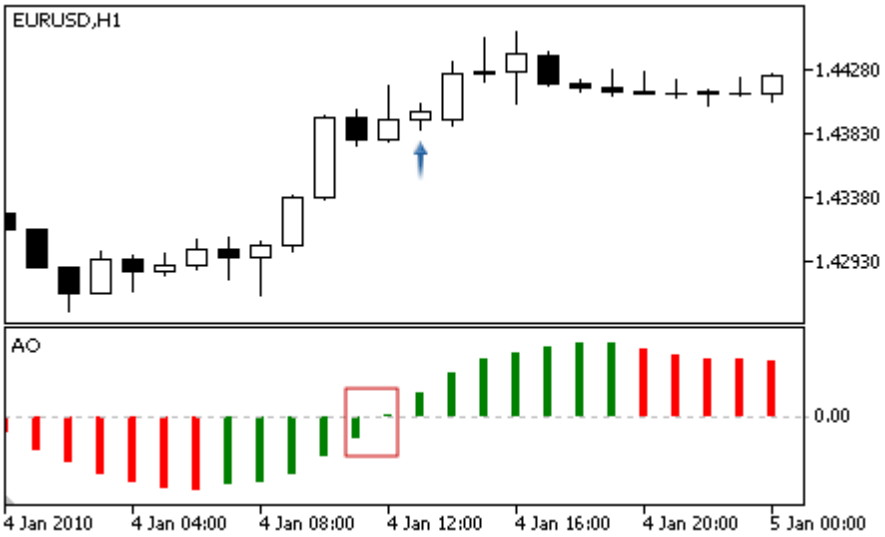
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Awesome Oscillator

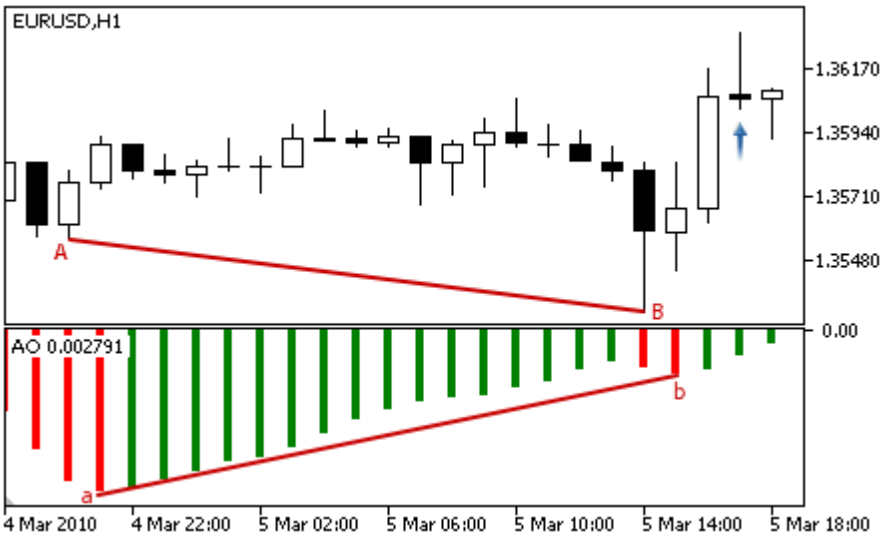
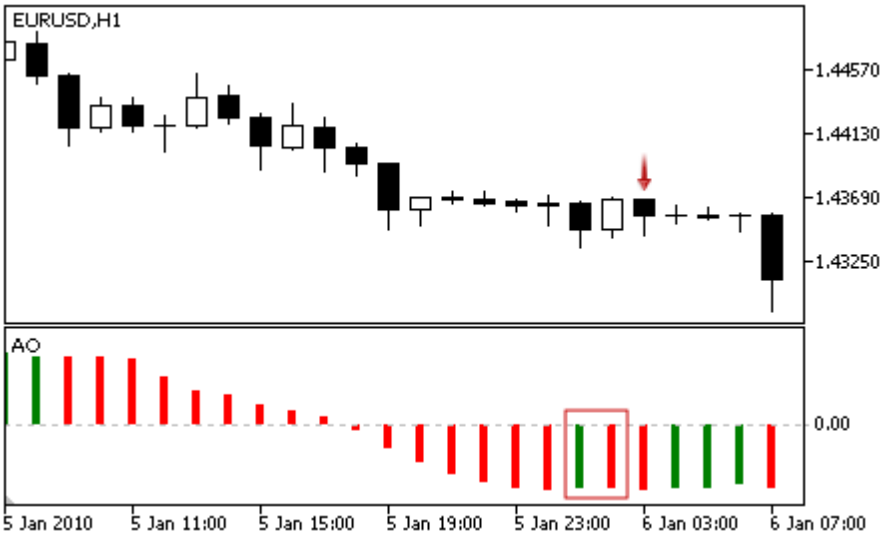
Le module des signaux basés sur les modèles de marché de l'indicateur Awesome Oscillator. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

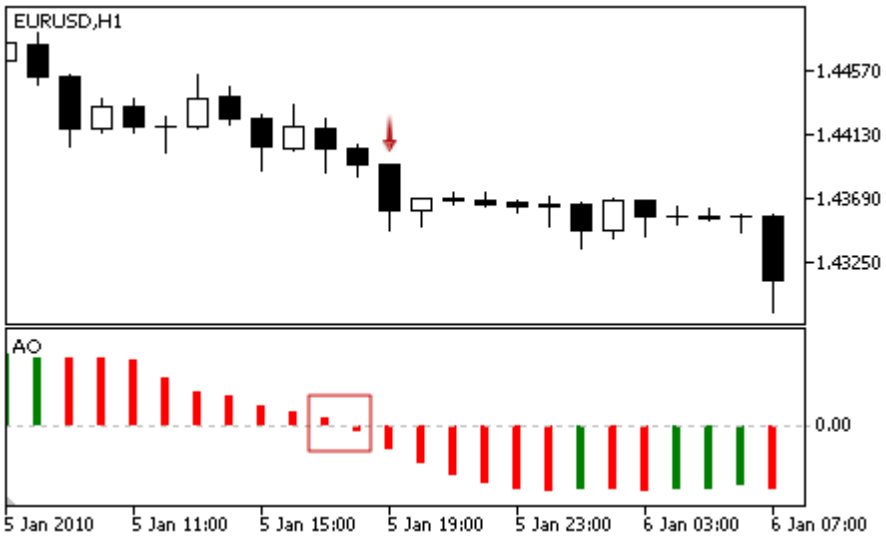
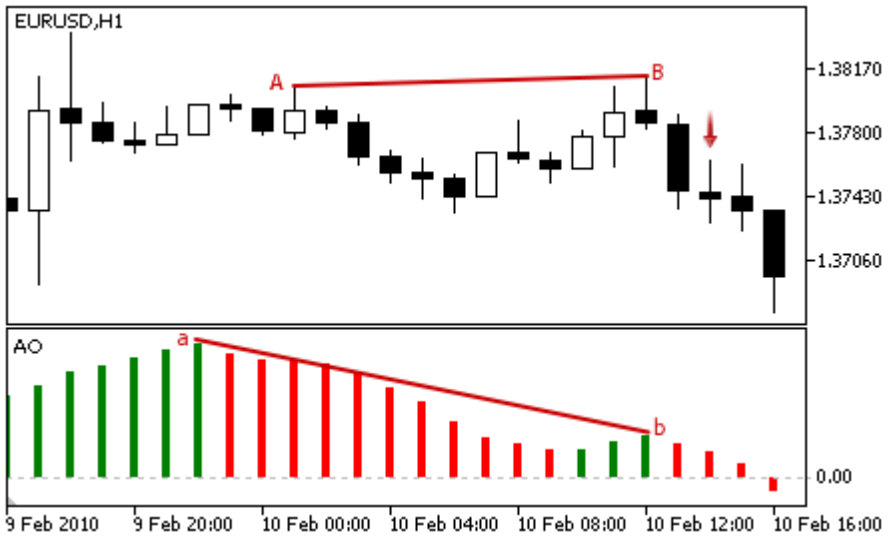
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Saucer** — la valeur de l'indicateur est en hausse sur la barre en cours d'analyse, alors qu'elle baissait sur la barre précédente ; les deux valeurs sont supérieure à 0.  • **Croisement de la ligne zéro** — la valeur de l'indicateur est positive sur la barre en cours d'analyse, et inférieure à 0 sur la barre précédente.  • **Divergence** — le premier creux analysé de l'indicateur est moins profond que le

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	précédent, et le creux correspondant des prix est plus profond que le précédent. De plus, l'indicateur ne doit pas repasser au-dessus de 0. 
For selling	• **Saucer** — la valeur de l'indicateur est en baisse sur la barre en cours d'analyse et grimpait sur la barre précédente, les deux valeurs sont inférieures à 0.  • **Croisement de la ligne 0** — la valeur de l'indicateur est inférieure à 0 sur la barre en cours d'analyse, et était supérieure à 0 sur la barre précédente.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	 • **Divergence** — le premier pic analysé de l'indicateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le précédent. De plus, l'indicateur ne doit pas passer sous la barre zéro. 
Aucune objection pour l'achat	La valeur de l'indicateur augmente sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

© 2000-2015, MetaQuotes Software Corp.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.

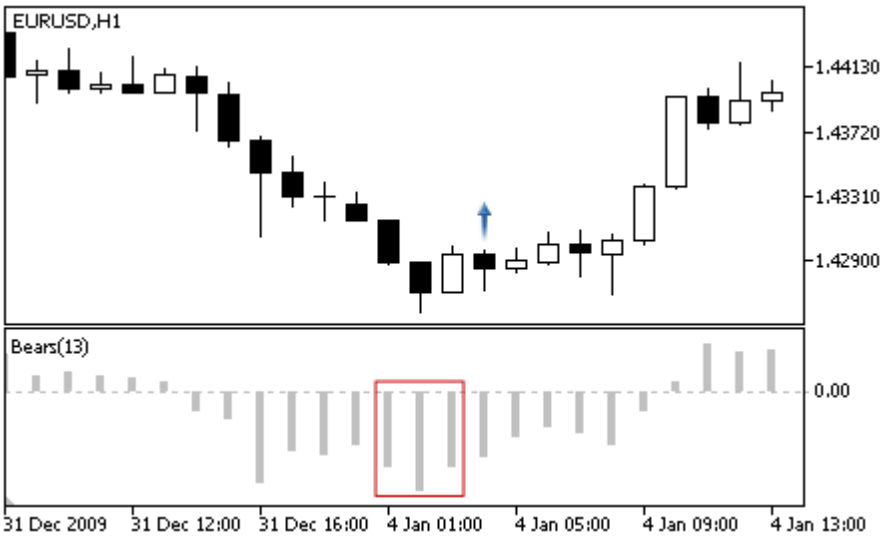
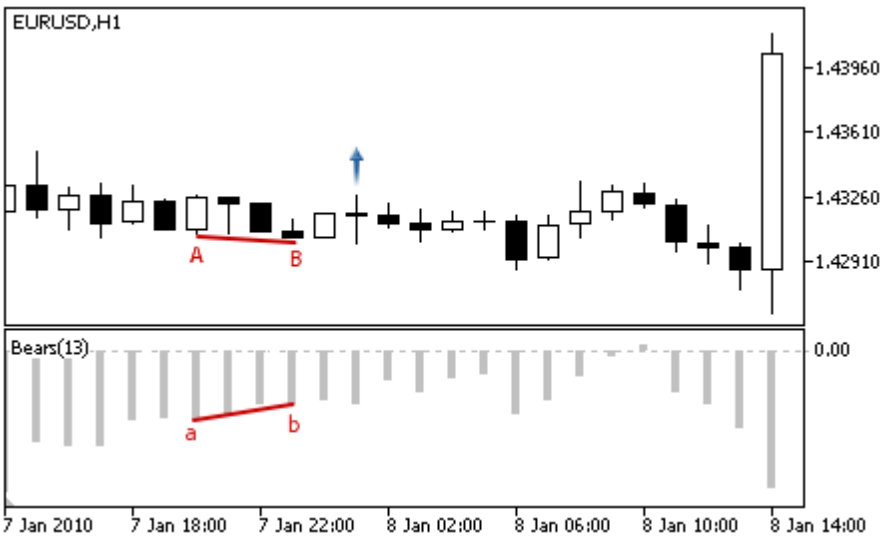
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Oscillateur Bears Power

Le module des signaux basés sur les modèles de marché de l'oscillateur Bears Power. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Renversement** — l'oscillateur s'est retourné à la hausse et sa valeur sur la barre en cours d'analyse est inférieure à 0.  • **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent. De plus, l'oscillateur ne doit grimper au-dessus de 0. 
For selling	Aucun signal de vente.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
Aucune objection pour l'achat	La valeur de l'oscillateur est inférieure à 0.
Aucune objection pour la vente	Aucun signal.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodBears	Période de calcul de l'oscillateur.

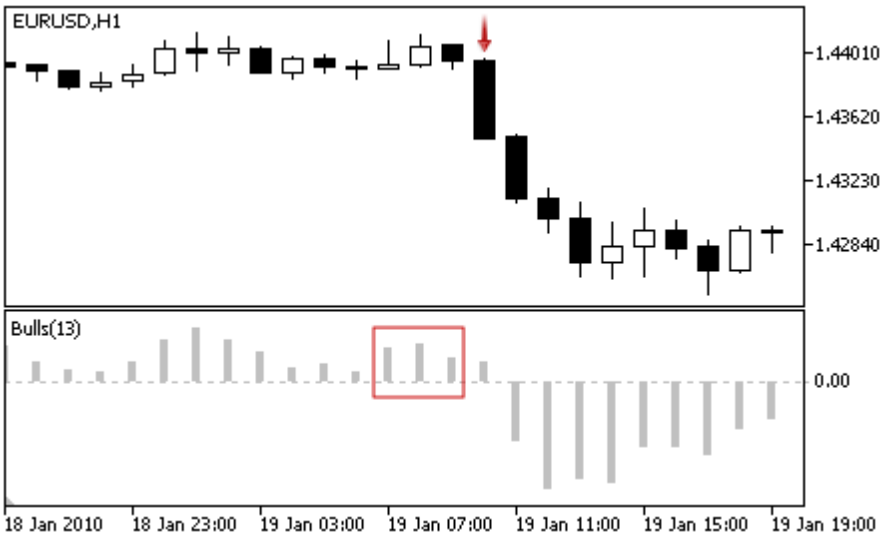
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Oscillateur Bulls Power

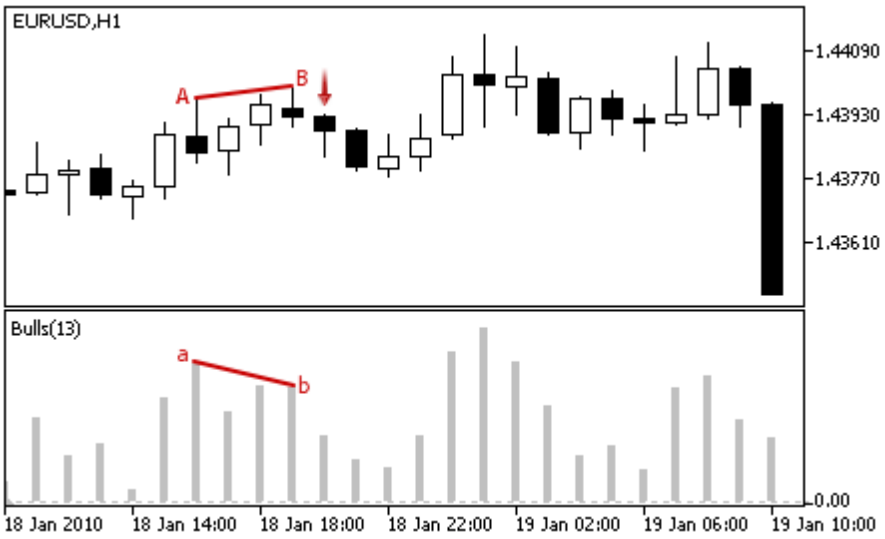
Ce module de signaux est basé sur les modèles de marché de l'oscillateur Bulls Power. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	Aucun signal d'achat.
For selling	• **Renversement** — l'oscillateur s'est retourné à la baisse et sa valeur sur la barre en cours d'analyse est supérieure à 0.  • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent. De plus, l'oscillateur ne doit pas chuter en dessous de 0.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	
Aucune objection pour l'achat	Aucun signal.
Aucune objection pour la vente	La valeur de l'oscillateur est supérieure à 0.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodBulls	Période de calcul de l'oscillateur.

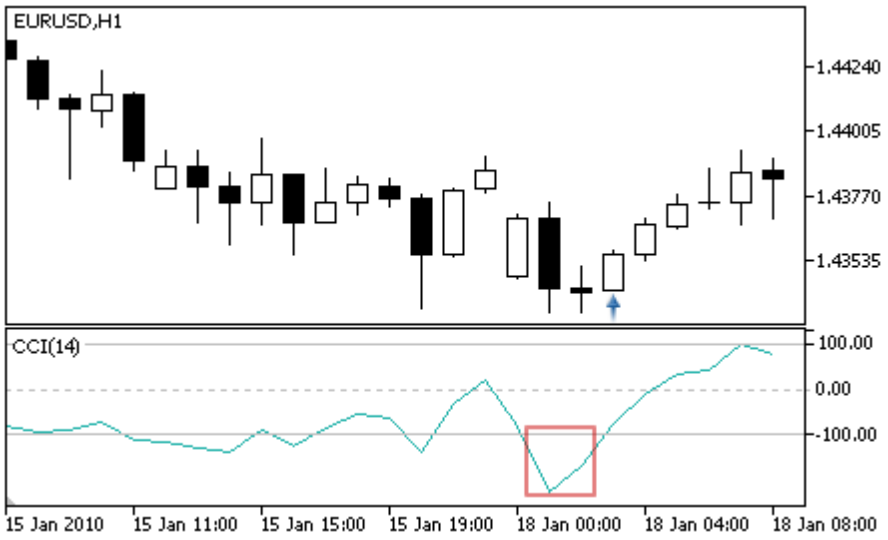
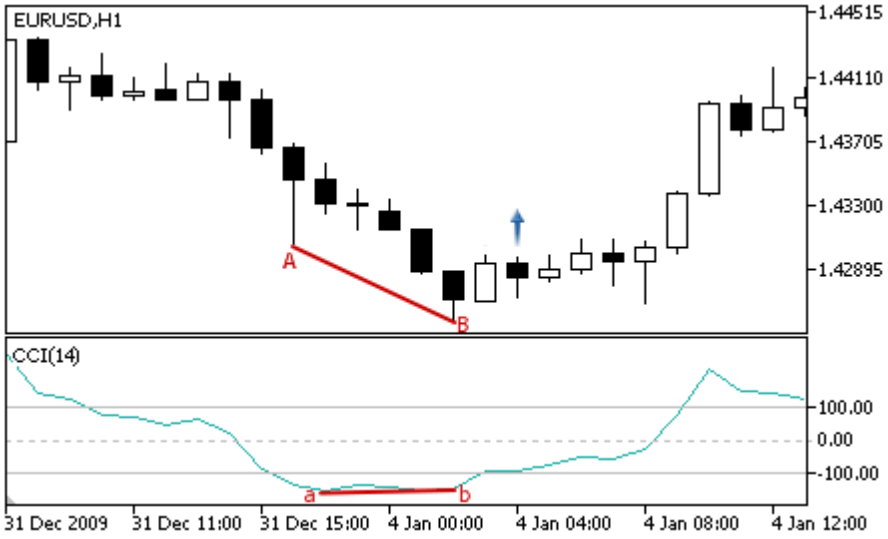
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Oscillateur Commodity Channel Index

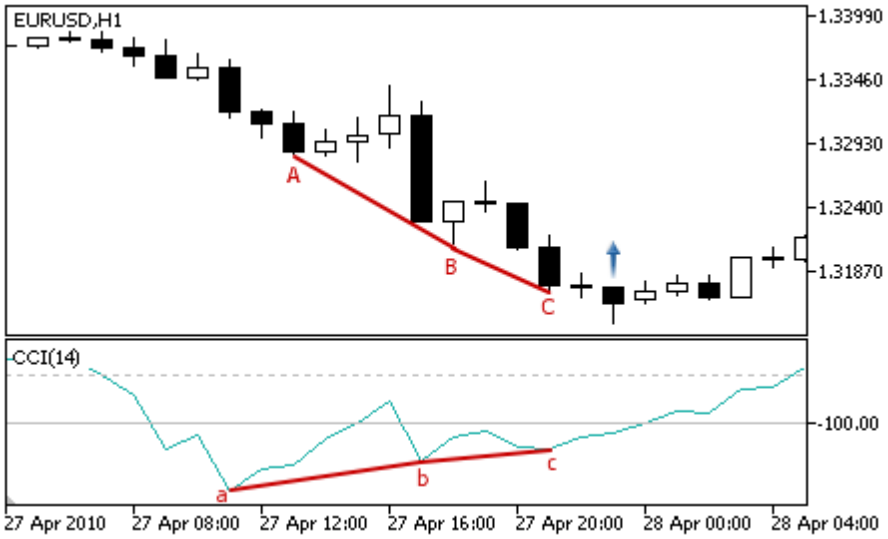
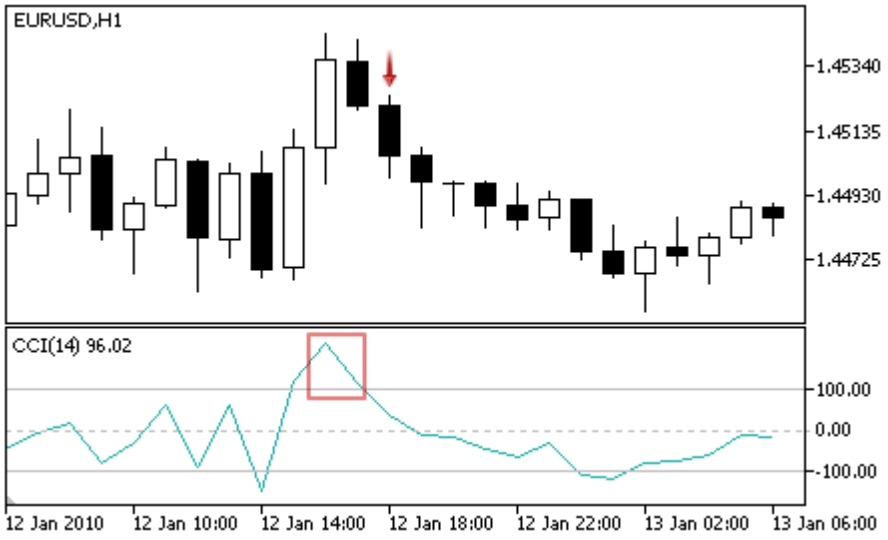
Ce module de signaux est basé sur les modèles de marché de l'oscillateur Commodity Channel Index. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

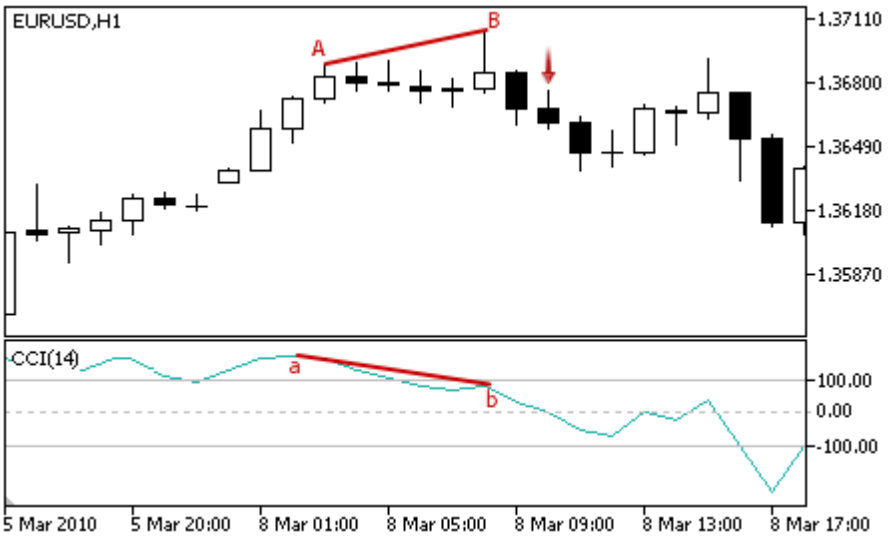
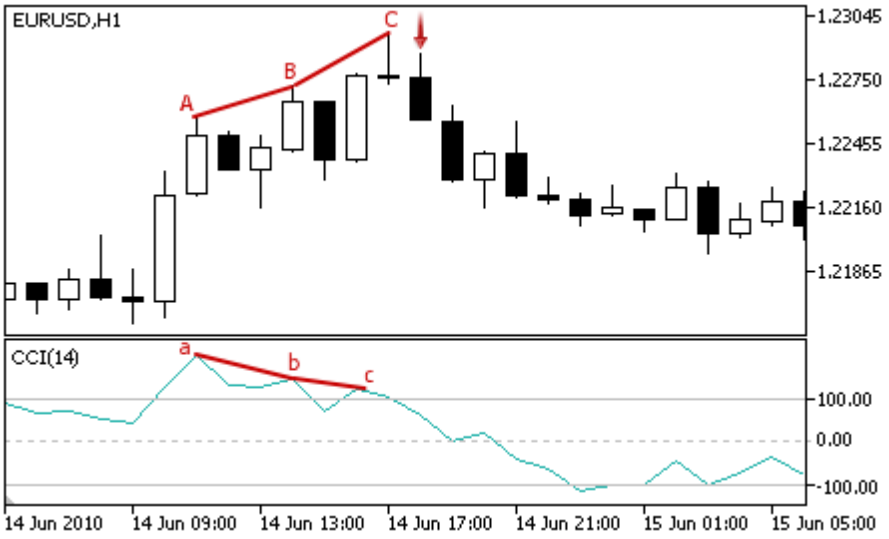
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Renversement en zone de sur-vente** — l'oscillateur se retourne à la hausse et sa valeur sur la barre en cours d'analyse est dans la zone de sur-vente (la valeur par défaut est -100).  • **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent. 

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	• **Double Divergence** — l'oscillateur forme 3 creux consécutifs, chacun d'eux est supérieur au précédent ; et les prix forment également les 3 creux correspondants, et chacun d'eux est plus bas que le précédent. 
For selling	• **Renversement en zone de sur-achat** — l'oscillateur se retourne à la baisse et sa valeur sur la barre en cours d'analyse est dans la zone de sur-achat (la valeur par défaut est 100).  • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	 • **Double Divergence** — l'oscillateur forme 3 pics consécutifs, chacun d'eux est inférieur au précédent ; et les prix forment également les 3 pics correspondants, et chacun d'eux est plus haut que le précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

© 2000-2015, MetaQuotes Software Corp.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodCCI	Période de calcul de l'oscillateur.
Applied	Une série de prix utilisée pour le calcul de l'oscillateur.

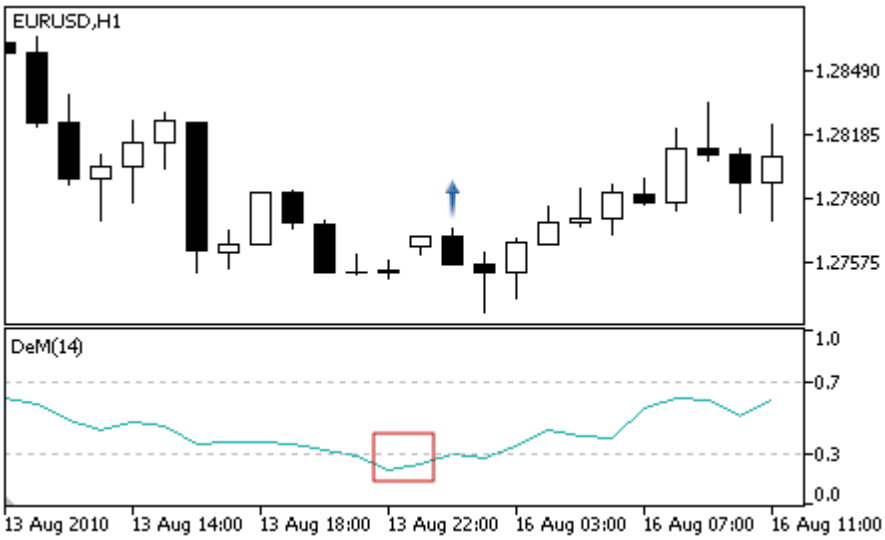
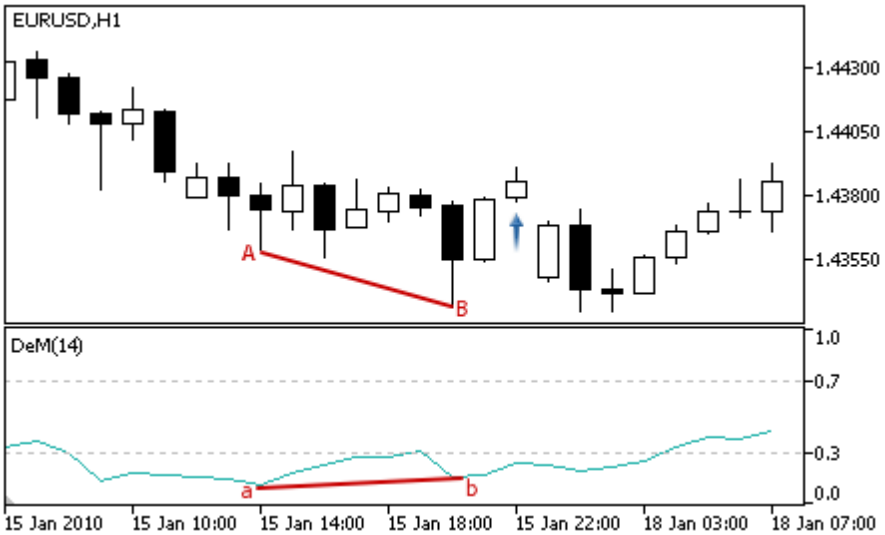
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Oscillateur DeMarker

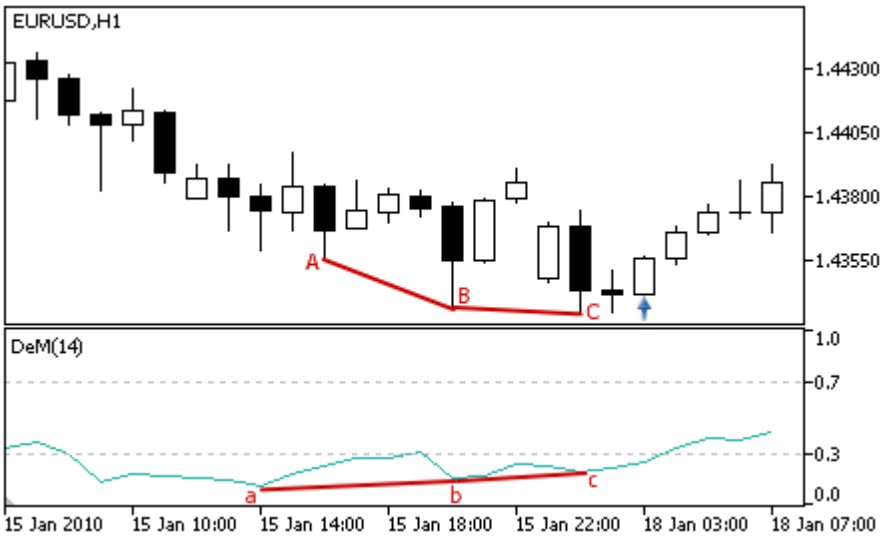
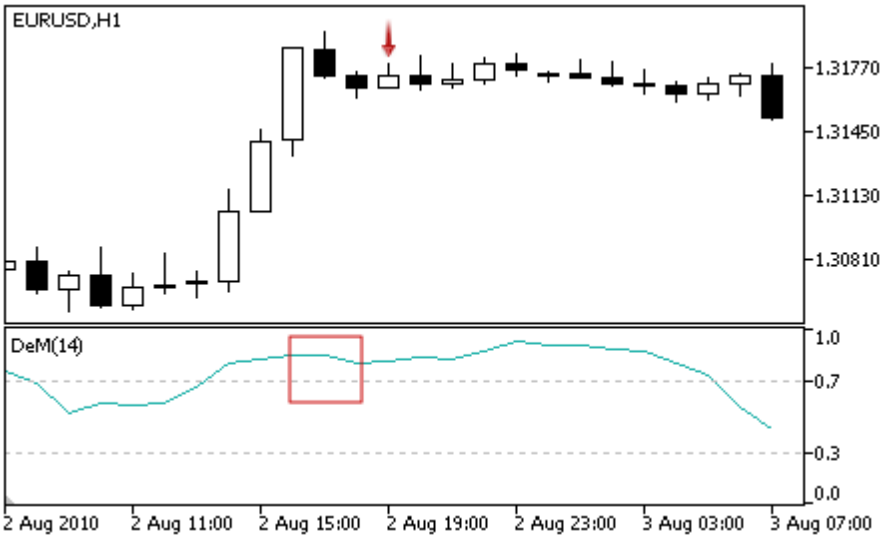
Ce module de signaux est basé sur les modèles de marché de l'oscillateur DeMarker. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

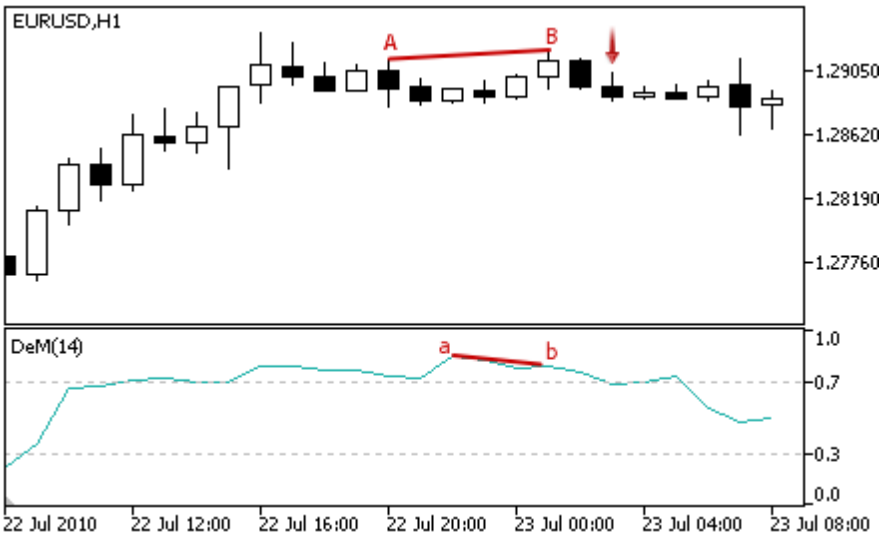
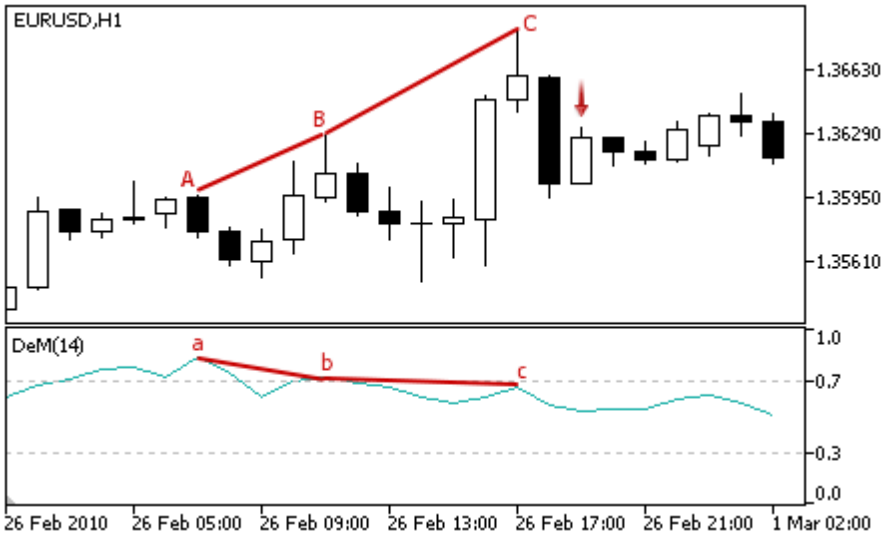
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Retournement dans la zone de sur-vente** — l'oscillateur se retourne à la hausse et sa valeur sur la barre analysée est dans la zone de sur-vente (la valeur par défaut est 0,3).  • **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent.  • **Double Divergence** — l'oscillateur forme 3 creux consécutifs, chacun d'eux est supérieur au précédent ; et les prix forment également les 3 creux correspondants,

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	et chacun d'eux est plus bas que le précédent. 
For selling	• **Renversement en zone de sur-achat** — l'oscillateur se retourne à la baisse et sa valeur sur la barre en cours d'analyse est dans la zone de sur-achat (la valeur par défaut est 0,7).  • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	 • **Double Divergence** — l'oscillateur forme 3 pics consécutifs, chacun d'eux est inférieur au précédent ; et les prix forment également les 3 pics correspondants, et chacun d'eux est plus haut que le précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

© 2000-2015, MetaQuotes Software Corp.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodDeM	Période de calcul de l'oscillateur.

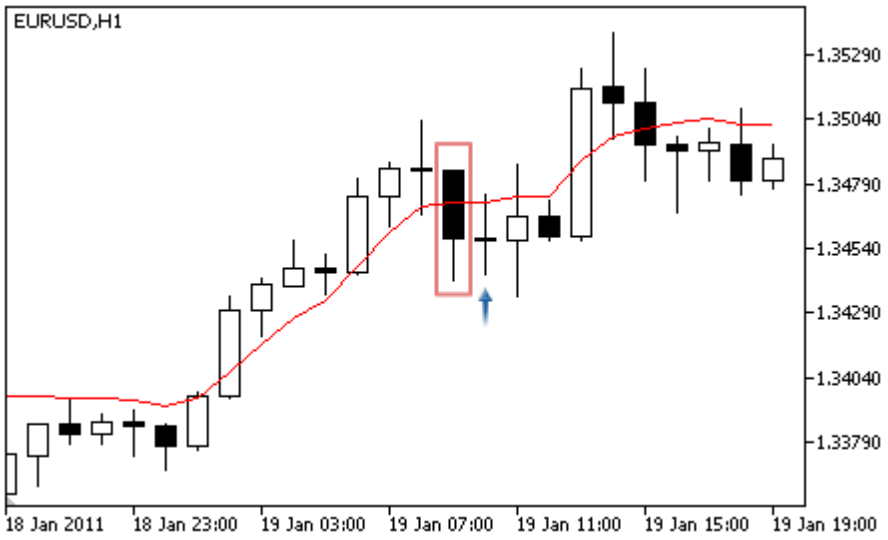
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Double Exponential Moving Average

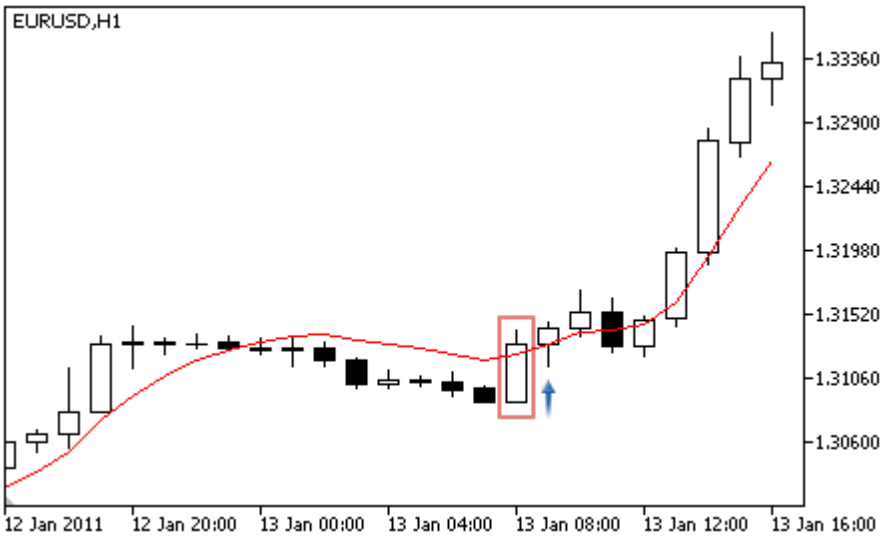
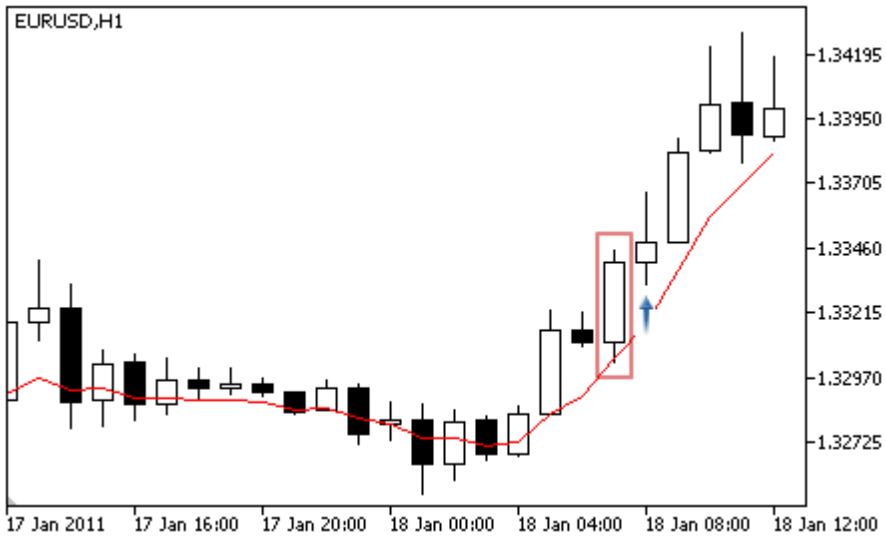
Ce module est basé sur les modèles de marché de l'indicateur >Double Exponential Moving Average (Moyenne Mobile Exponentielle Double). Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

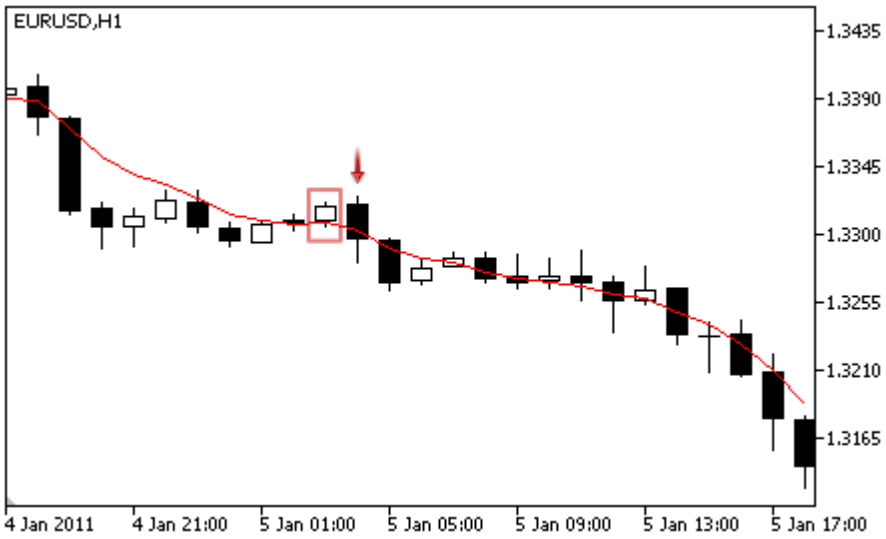
Type de Signal	Description des Conditions
Pour l'achat	• Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en hausse (signal fort).

© 2000-2015, MetaQuotes Software Corp.

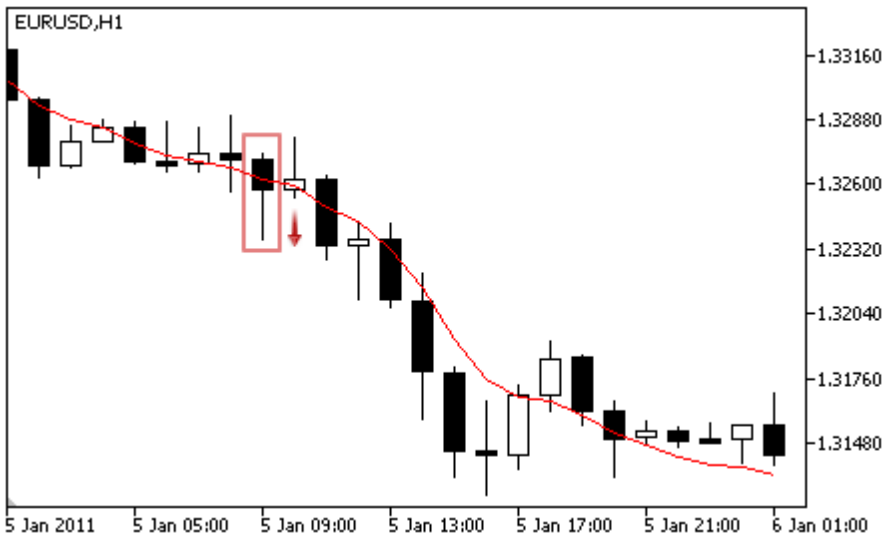
Type de Signal	Description des Conditions
	 • L'ombre inférieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont au-dessus de l'indicateur, et le prix le plus bas est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible). 
For selling	• Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions

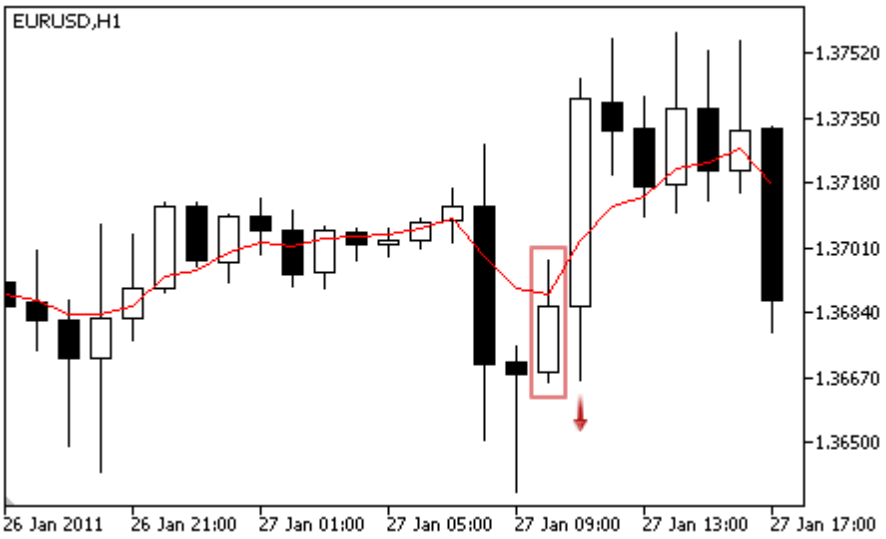


- **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en baisse (signal fort).



- L'ombre supérieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont en-dessous de l'indicateur, et le prix le plus haut est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	
Aucune objection pour l'achat	Les prix sont au-dessus de l'indicateur.
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de lissage de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

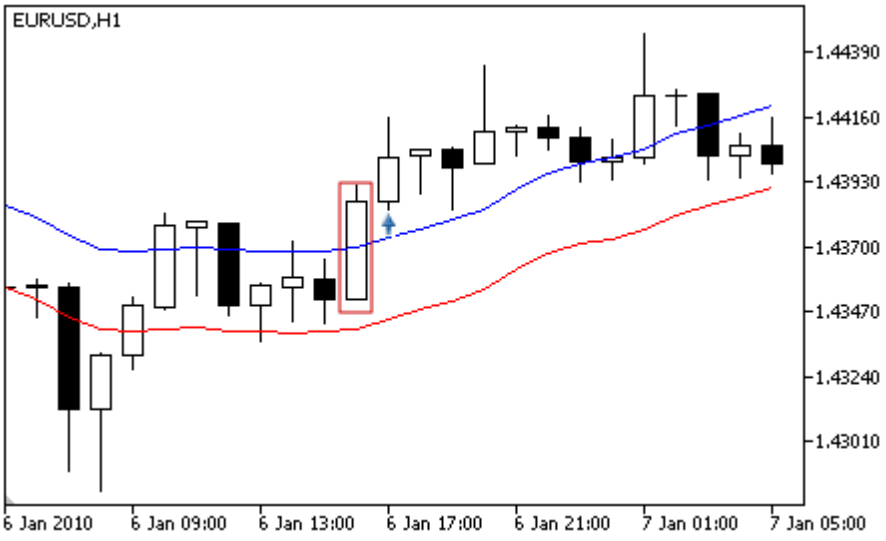
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Envelopes

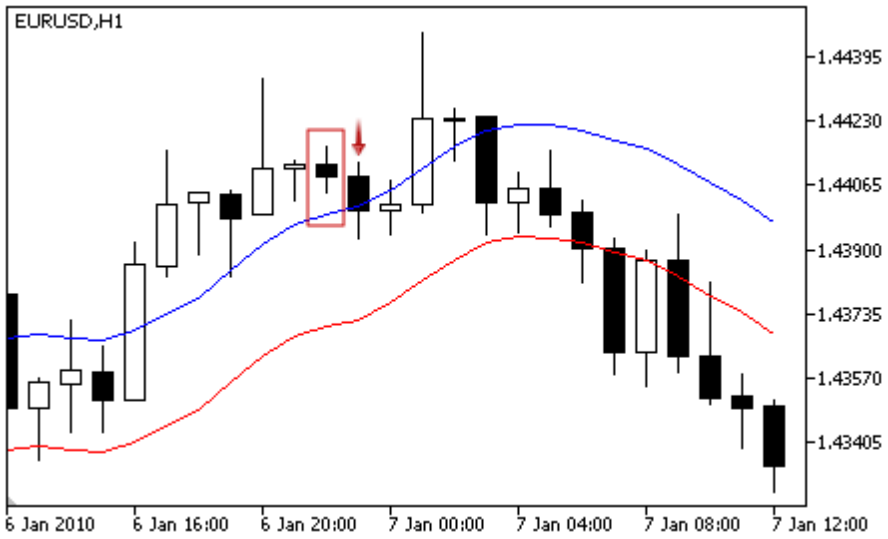
Ce module de signaux est basé sur les modèles de marché de l'indicateur Awesome Envelopes. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• Le prix est près de la ligne inférieure de l'indicateur sur la barre analysée.  • Le prix a croisé la ligne supérieure de l'indicateur sur la barre analysée. 
For selling	• Le prix est près de la ligne supérieure de l'indicateur sur la barre analysée.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	 • Le prix a croisé la ligne inférieure de l'indicateur sur la barre analysée. 
Aucune objection pour l'achat	Aucun signal.
Aucune objection pour la vente	Aucun signal.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

© 2000-2015, MetaQuotes Software Corp.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de calcul de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.
Deviation	Déviation des bordures de l'enveloppe depuis la ligne centrale (la moyenne mobile) en pourcentages.

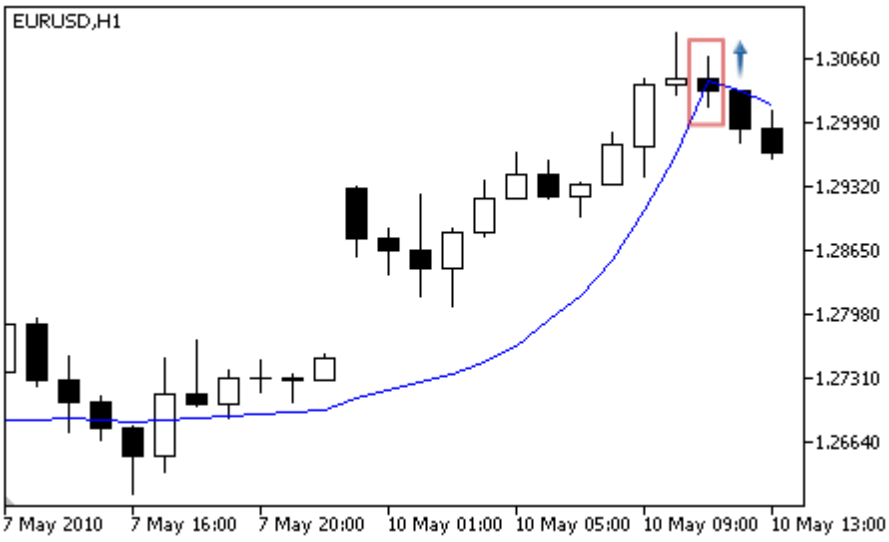
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Fractal Adaptive Moving Average

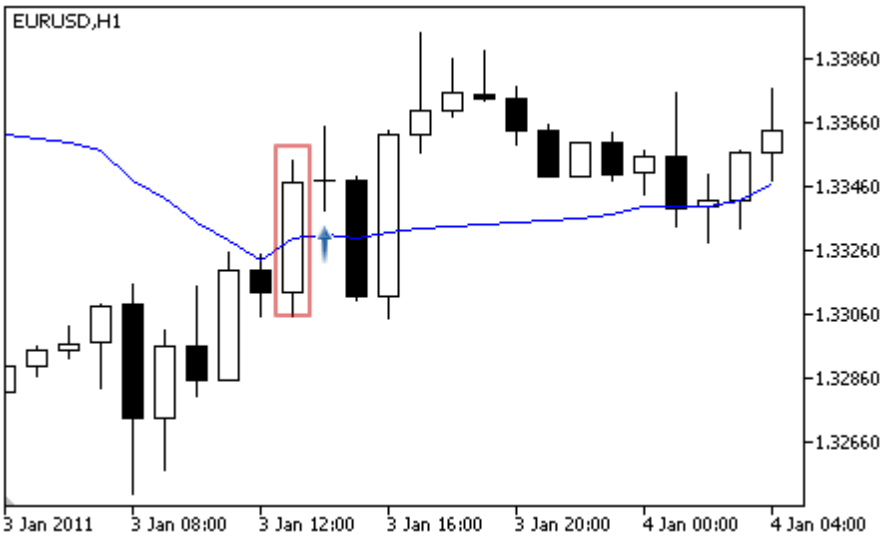
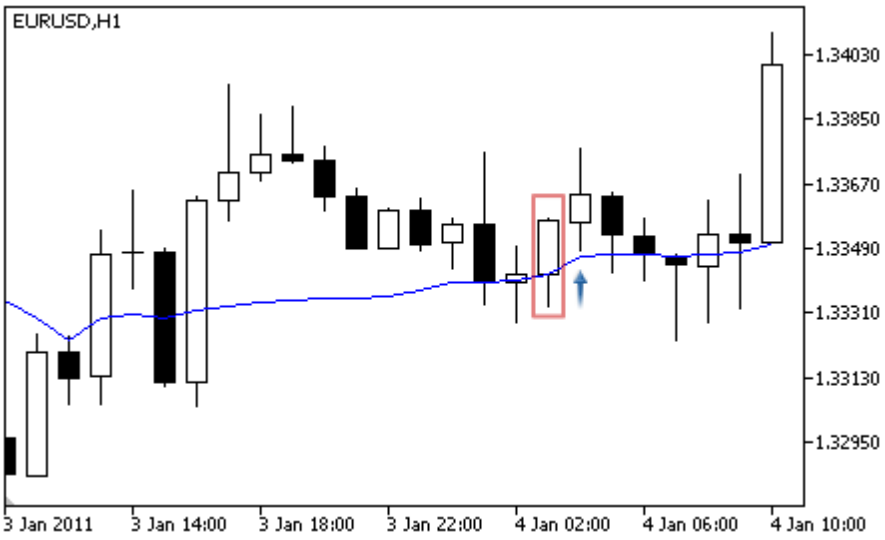
Ce module de signaux est basé sur les modèles de marché de l'indicateurFractal Adaptive Moving Average. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

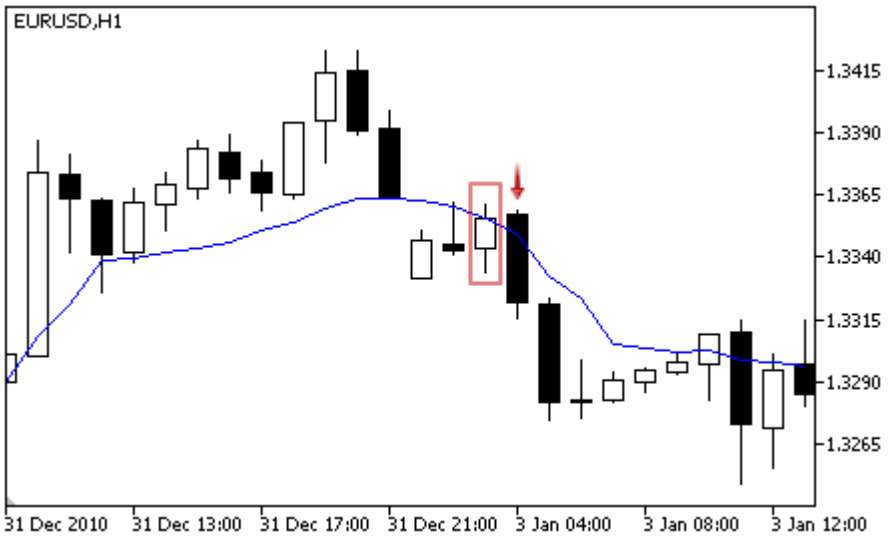
Type de Signal	Description des Conditions
Pour l'achat	• Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en hausse (signal fort).

© 2000-2015, MetaQuotes Software Corp.

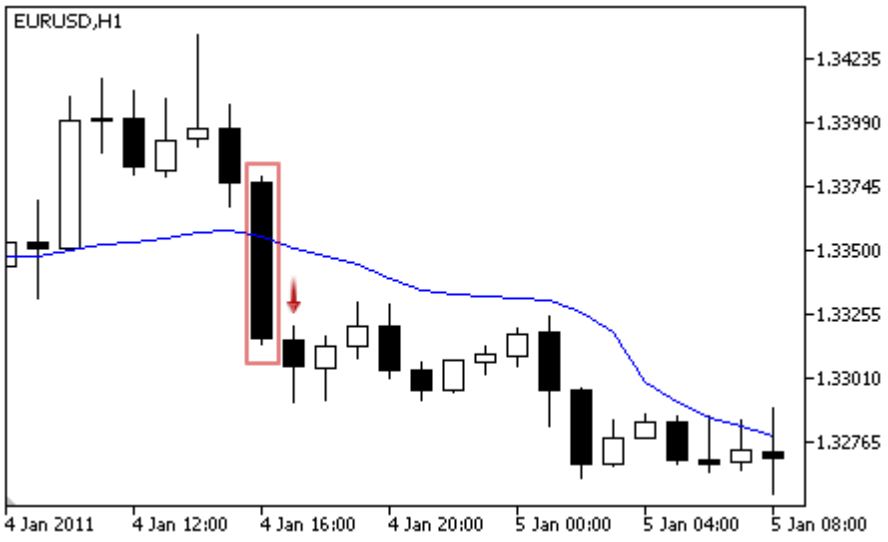
Type de Signal	Description des Conditions
	 • L'ombre inférieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont au-dessus de l'indicateur, et le prix le plus bas est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible). 
For selling	• Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions

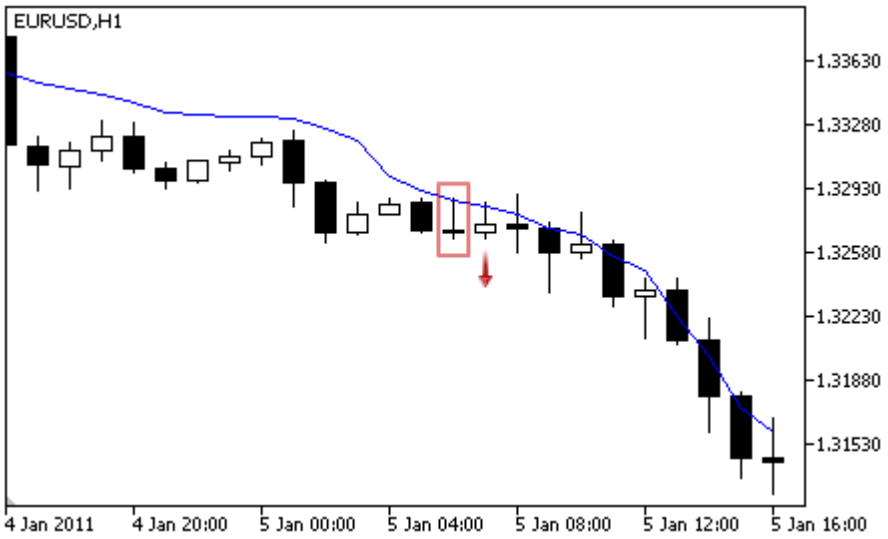


- **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en baisse (signal fort).



- L'ombre supérieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont en-dessous de l'indicateur, et le prix le plus haut est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	
Aucune objection pour l'achat	Les prix sont au-dessus de l'indicateur.
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de lissage de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

Signaux du Filtre Horaire Intraday

Ce module est basé sur l'assomption que l'efficience des modèles de marché change avec le temps. En utilisant ce module, vous pouvez filtrer les signaux, reçus d'autres modules, par heures et jours de la semaine. Cela permet d'améliorer la qualité des signaux générés grâce à la suppression des périodes de temps les moins favorables. Ce mécanisme de prise de décisions sur la base de signaux des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	Aucun signal.
For selling	Aucun signal.
Aucune objection pour l'achat	L'heure et la date actuelles correspondent aux paramètres spécifiés.
Aucune objection pour la vente	L'heure et la date actuelles correspondent aux paramètres spécifiés.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
GoodHourOfDay	Numéro de l'heure du jour (de 0 à 23) où les signaux seront activés. Si la valeur est -1, les signaux seront activés pour la journée entière.
BadHoursOfDay	Le champ bit. Chaque bit de ce champ correspond à une heure de la journée (bit 0 - heure 0, ..., bit 23 - 23ème heure). Si la valeur d'un bit est égale à 0, les signaux de trading seront activés durant l'heure correspondante. Si la valeur d'un bit est égale à 1, les signaux de trading seront désactivés durant l'heure correspondante. Un nombre spécifié est représenté sous la forme d'un nombre binaire et est utilisé comme un masque de bit. Les heures désactivées ont une plus grande priorité que les heures activées.
GoodDayOfWeek	Numéro du jours de la semaine (de 0 à 6 où 0

© 2000-2015, MetaQuotes Software Corp.

Paramètre	Description
	est dimanche) où les signaux seront activés. Si la valeur est -1, les signaux seront activés pour tous les jours.
BadDaysOfWeek	Le champ bit. Chaque bit de ce champ correspond à un jour de la semaine (bit 0 - dimanche, ..., bit 6 - samedi). Si la valeur d'un bit est égale à 0, les signaux de trading seront activés durant le jour correspondant. Si la valeur d'un bit est égale à 1, les signaux de trading seront désactivés durant la journée correspondante. Un nombre spécifié est représenté sous la forme d'un nombre binaire et est utilisé comme un masque de bit. Les jours désactivés ont une plus grande priorité que les jours activés.

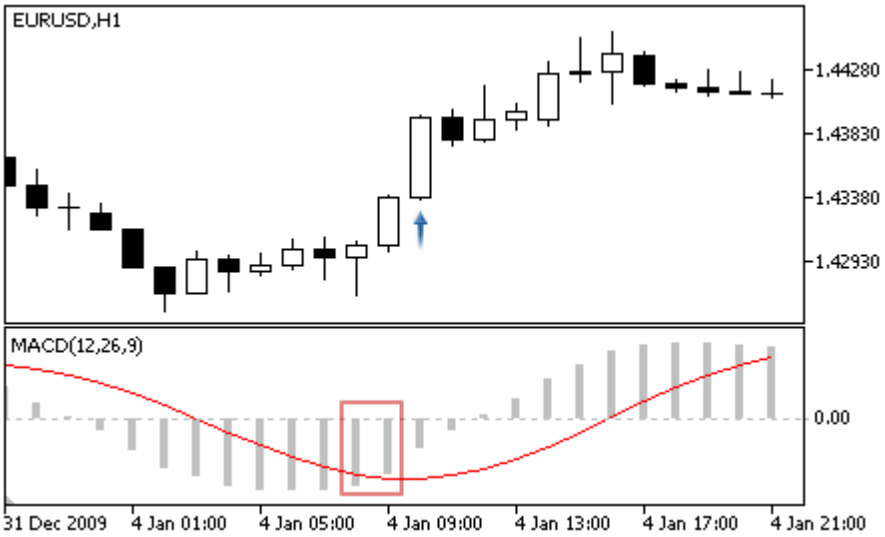
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'oscillateur MACD

Ce module de signaux est basé sur les modèles de marché de l'oscillateur MACD. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

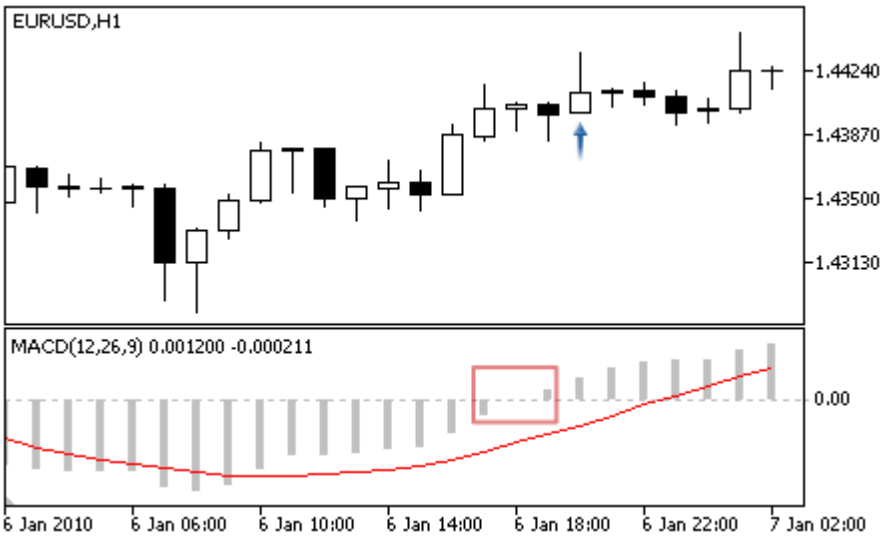
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Renversement** — l'oscillateur s'est retourné à la hausse (l'oscillateur monte sur la barre analysée et baisse sur la barre précédente).  • **Croisement de la ligne de macd et de la ligne de signal** — la ligne de macd est au-dessus de la ligne du signal sur la barre analysée et sous la ligne de signal sur la barre précédente.  • **Croisement avec la ligne zéro** — la ligne de macd est au-dessus de zéro sur la

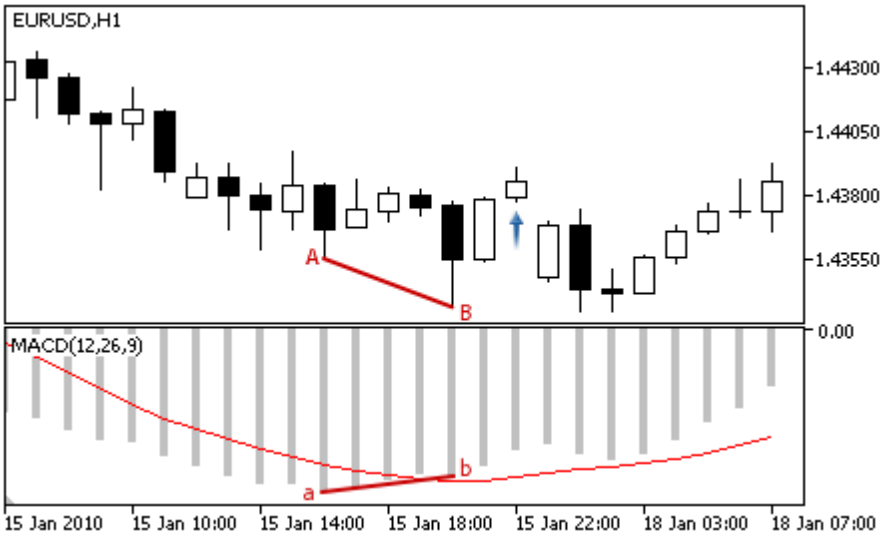
© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions

barre analysée et sous la ligne de zéro sur la barre précédente.

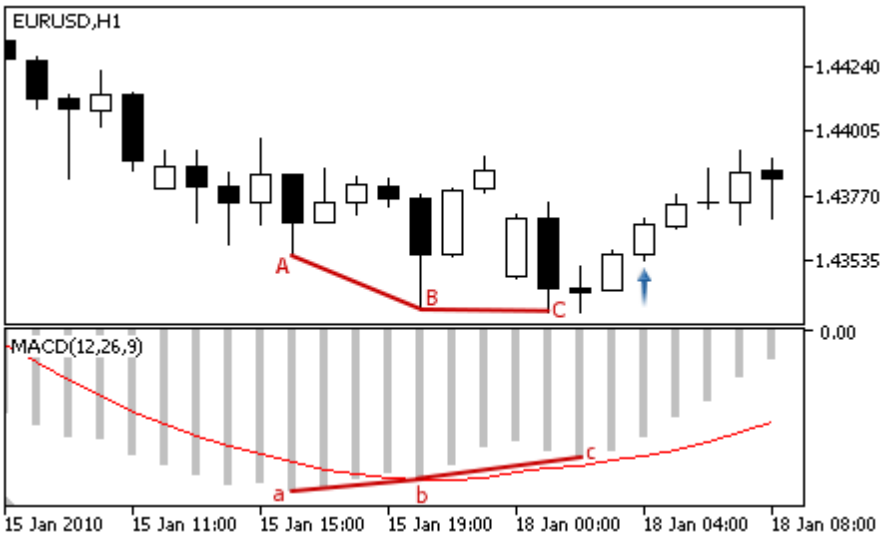
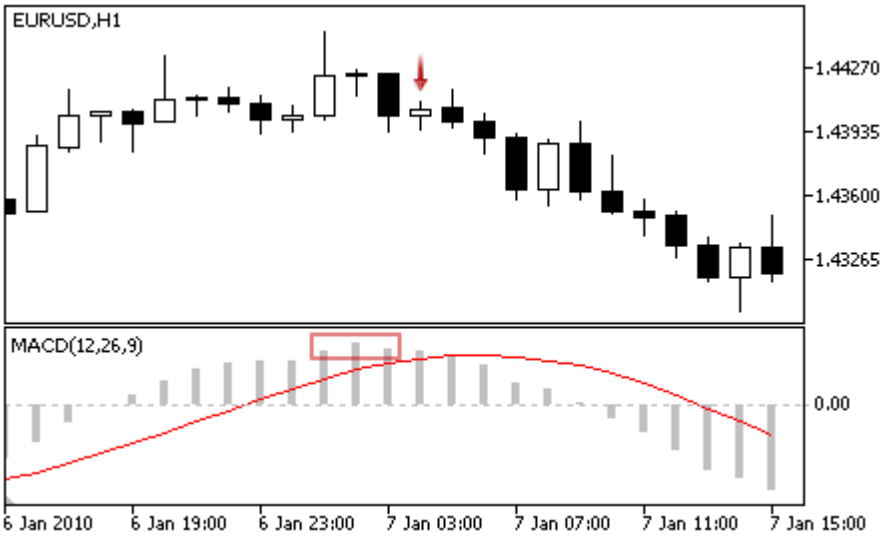


- **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent.



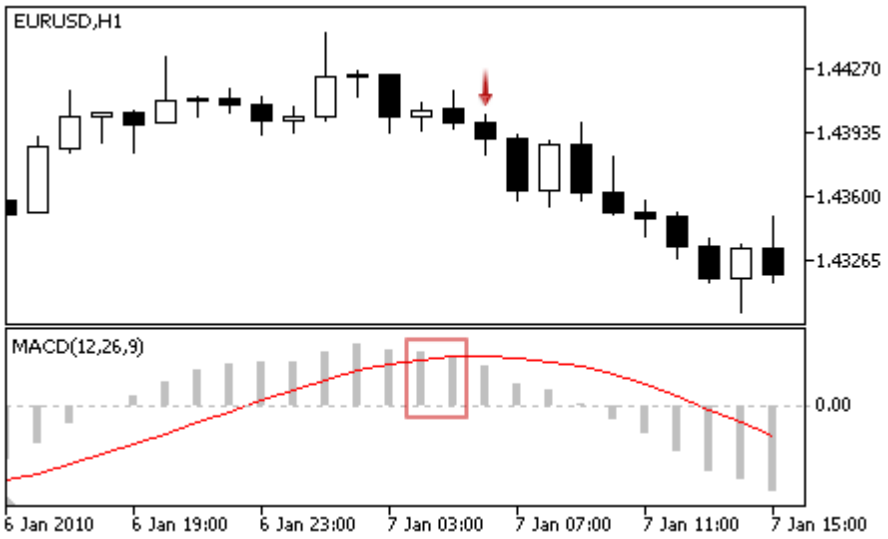
- **Double Divergence** — l'oscillateur forme 3 creux consécutifs, chacun d'eux est supérieur au précédent ; et les prix forment également les 3 creux correspondants, et chacun d'eux est plus bas que le précédent.

© 2000-2015, MetaQuotes Software Corp.

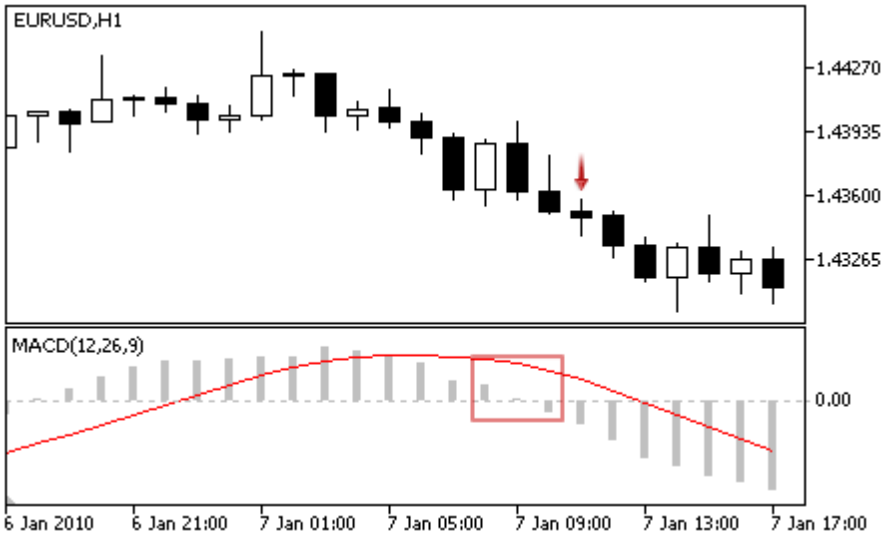
Type de Signal	Description des Conditions
	
For selling	• **Renversement** — l'oscillateur se retourne à la baisse (l'oscillateur est en baisse sur la barre analysée et en hausse sur la barre précédente).  • **Croisement de la ligne de macd et de la ligne de signal** — la ligne de macd est en-dessous de la ligne du signal sur la barre analysée et au-dessus de la ligne de signal sur la barre précédente.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions

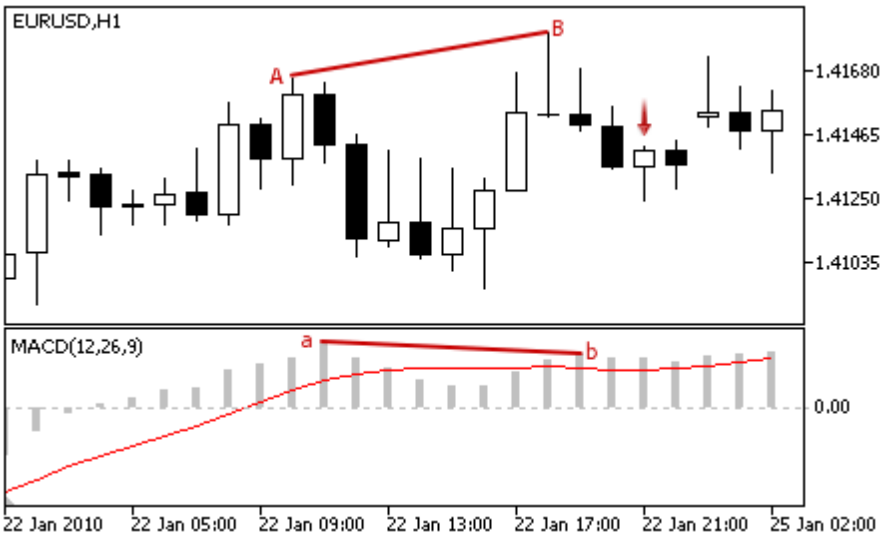
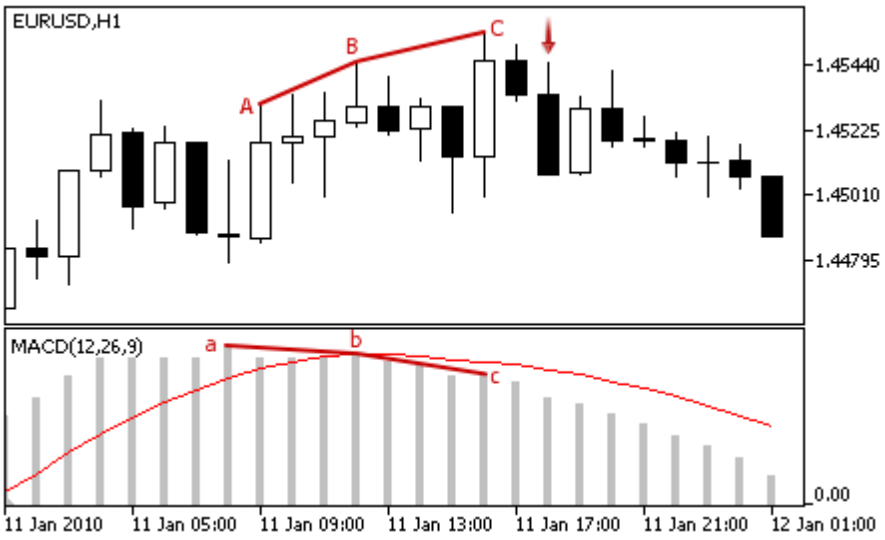


- **Croisement avec la ligne zéro** — la ligne de macd est en-dessous de zéro sur la barre analysée et au-dessus de la ligne de zéro sur la barre précédente.



- **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	 • **Double Divergence** — l'oscillateur forme 3 pics consécutifs, chacun d'eux est inférieur au précédent ; et les prix forment également les 3 pics correspondants, et chacun d'eux est plus haut que le précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

© 2000-2015, MetaQuotes Software Corp.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodFast	Période de calcul de la moyenne mobile rapide (fast EMA).
PeriodSlow	Période de calcul de la moyenne mobile lente (slow EMA).
PeriodSignal	Période de lissage.
Applied	Une série de prix utilisée pour le calcul de l'oscillateur.

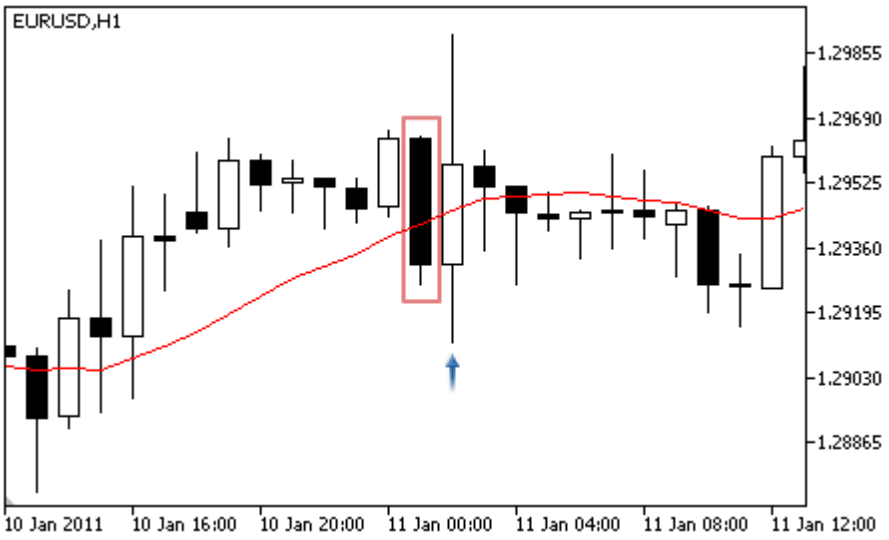
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Moving Average

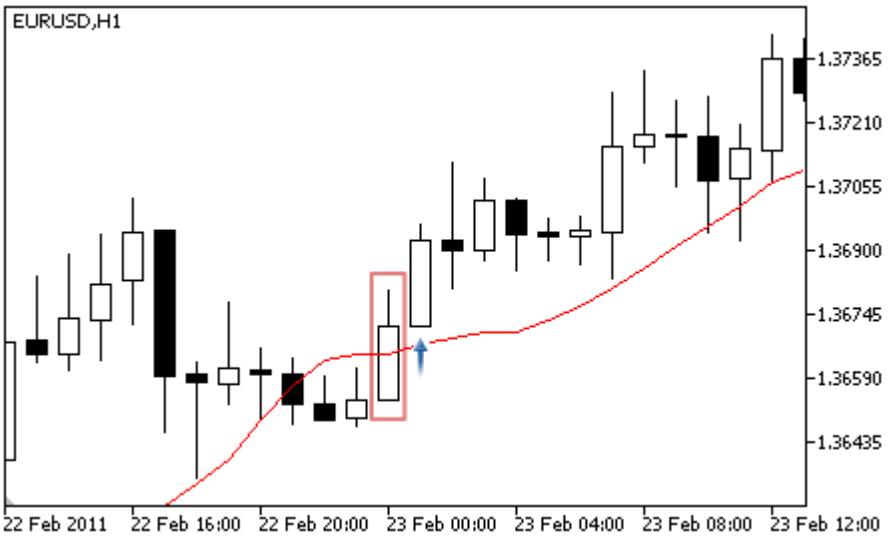
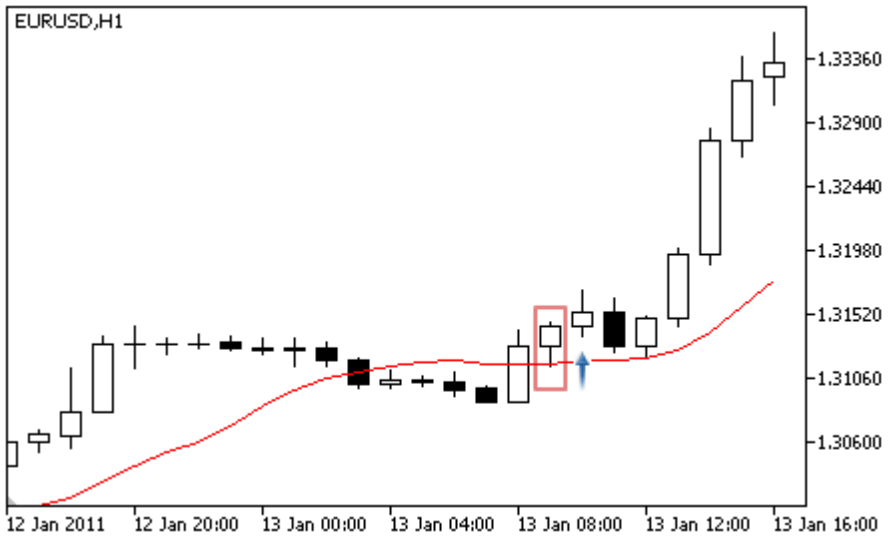
Ce module de signaux est basé sur les modèles de marché de l'indicateurMoyenne Mobile. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

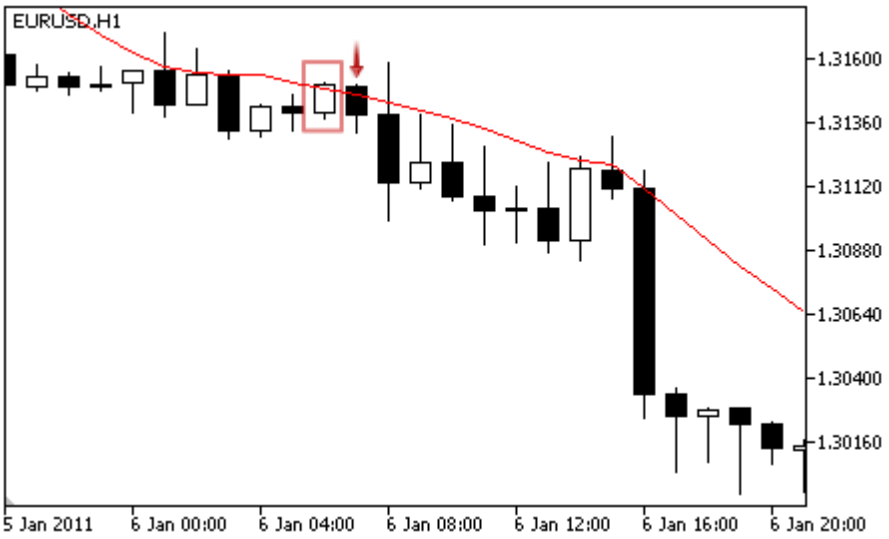
Type de Signal	Description des Conditions
Pour l'achat	• Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en hausse (signal fort).

© 2000-2015, MetaQuotes Software Corp.

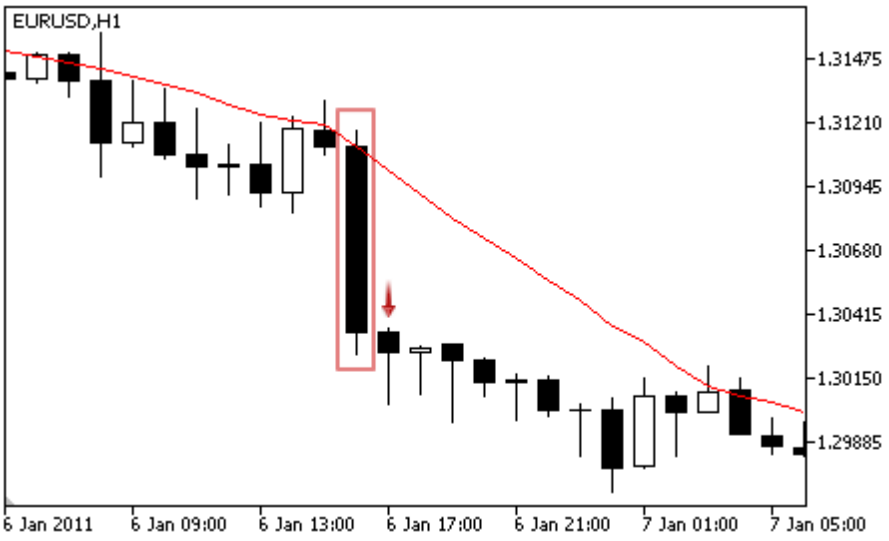
Type de Signal	Description des Conditions
	 • L'ombre inférieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont au-dessus de l'indicateur, et le prix le plus bas est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible). 
For selling	• Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions

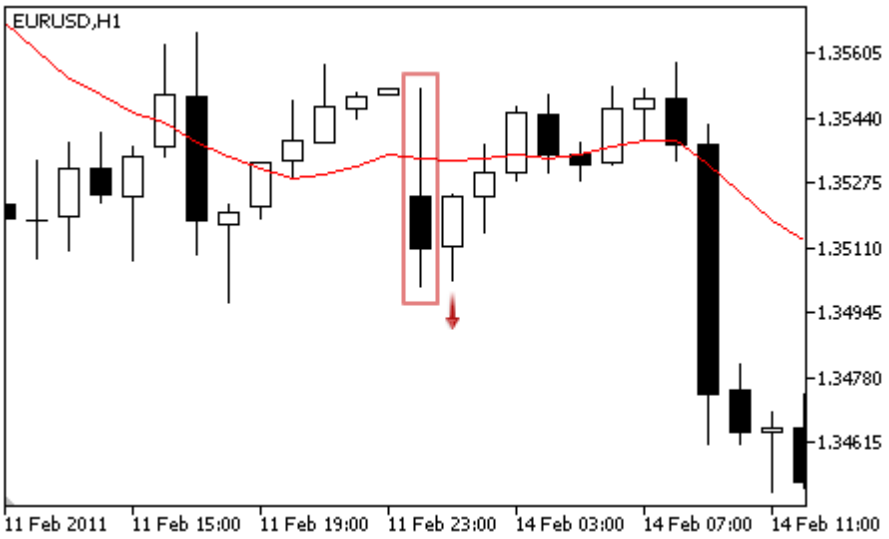


- **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en baisse (signal fort).



- L'ombre supérieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont en-dessous de l'indicateur, et le prix le plus haut est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	
Aucune objection pour l'achat	Les prix sont au-dessus de l'indicateur.
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de lissage de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

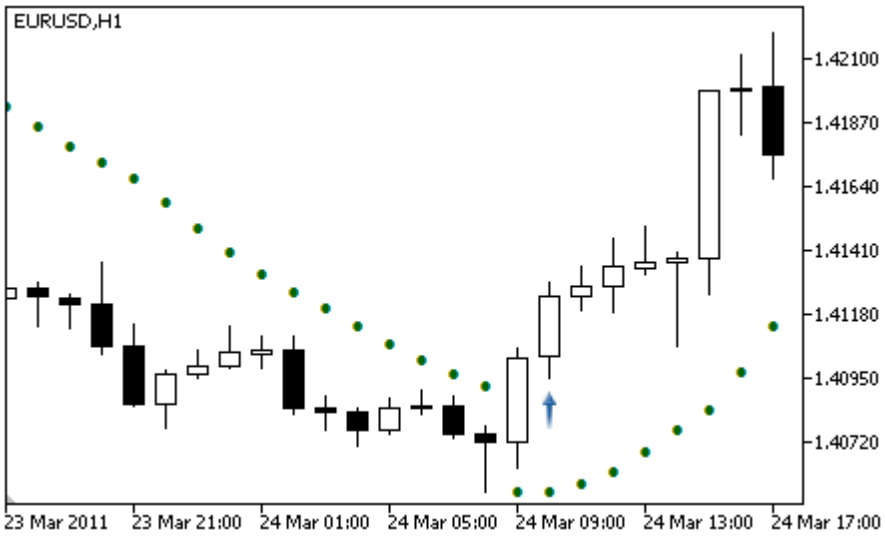
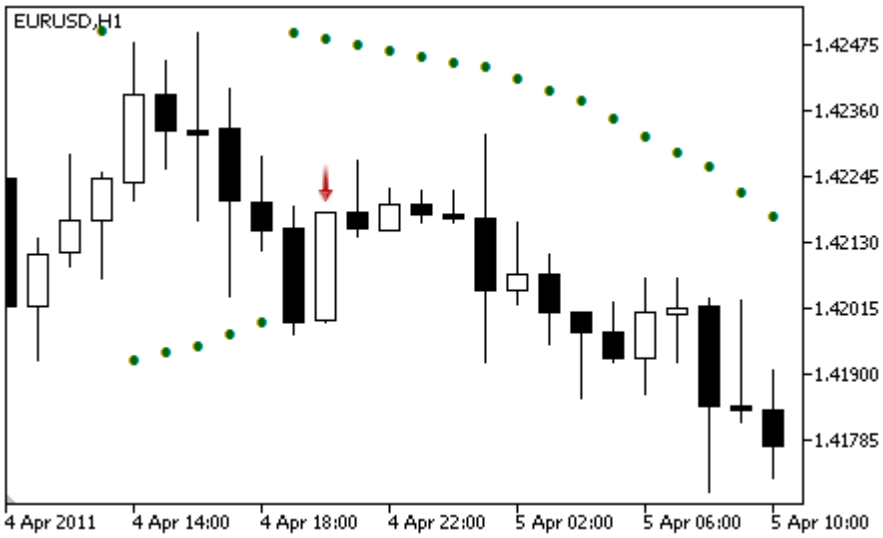
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Parabolic SAR

Ce module de signaux est basé sur les modèles de marché de l'indicateurParabolic SAR. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	**Renversement** — l'indicateur est sous le prix sur la barre analysée et au-dessus du prix sur la barre précédente. 
For selling	**Renversement** — l'indicateur est au-dessus du prix et sous le prix sur la barre précédente. 
Aucune objection	Les prix sont au-dessus de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
pour l'achat	
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
Step	L'incrément de vitesse de l'indicateur.
Maximum	Taux maximum de la vitesse de convergence de l'indicateur avec le prix.

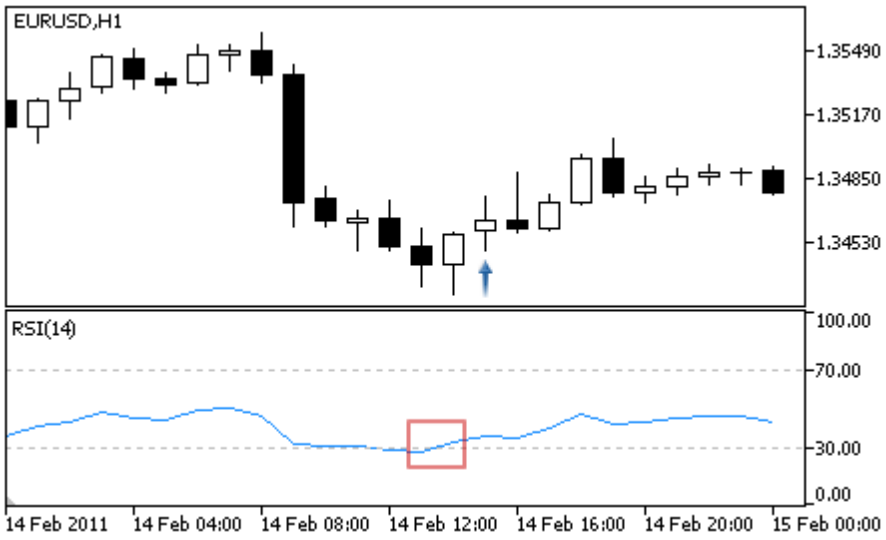
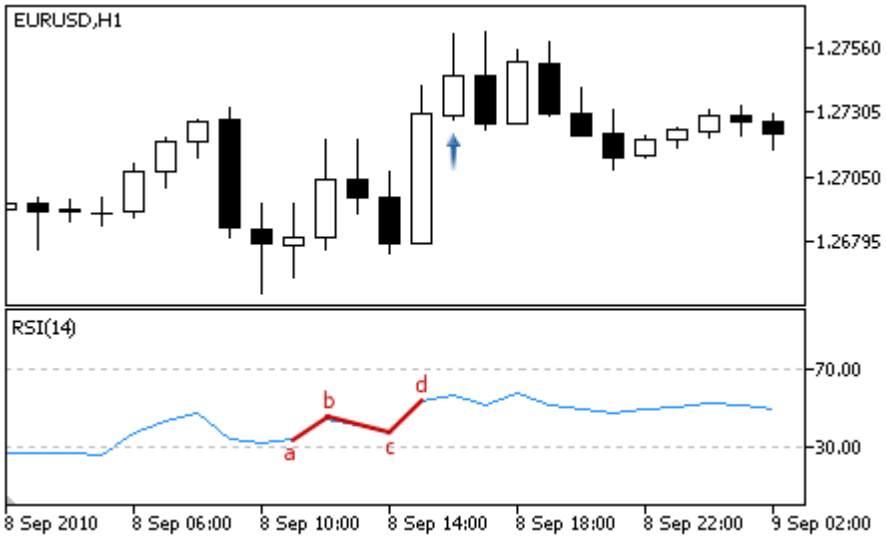
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'oscillateur Relative Strength Index

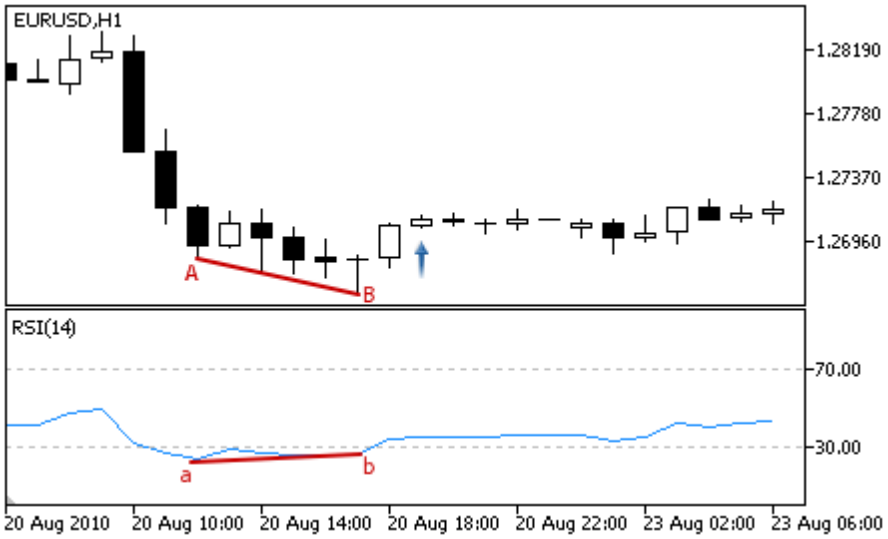
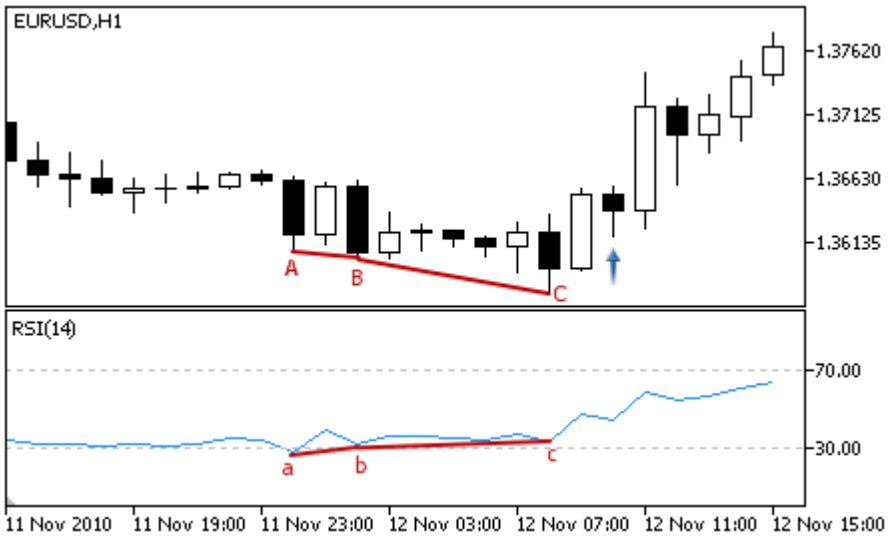
Ce module de signaux est basé sur les modèles de marché de l'oscillateur Relative Strength Index. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

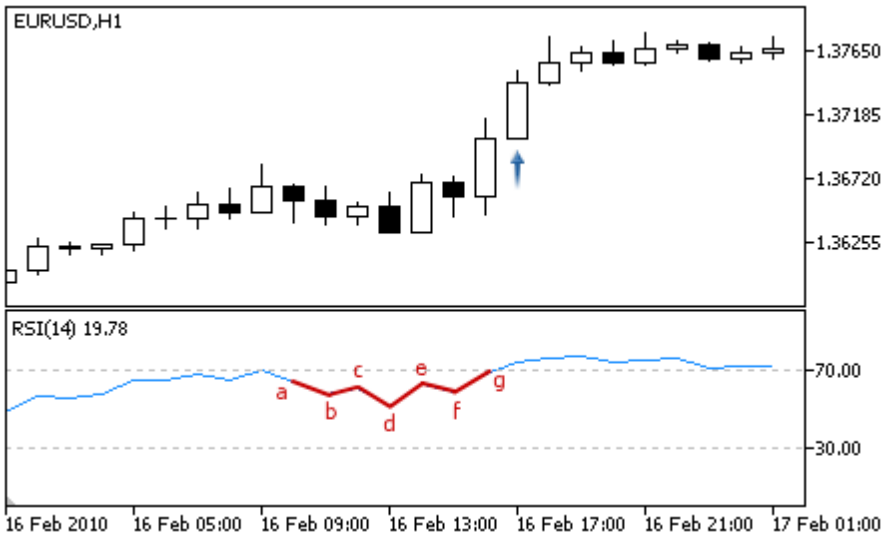
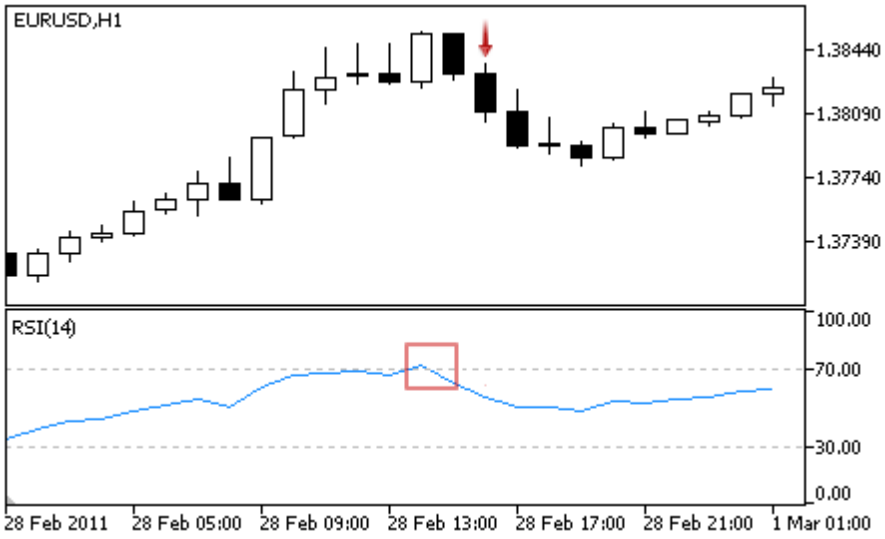
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Retournement dans la zone de sur-vente** — l'oscillateur se retourne à la hausse et sa valeur sur la barre analysée est dans la zone de sur-vente (la valeur par défaut est 30).  • **Failed swing** — l'oscillator monte plus haut que le pic précédent sur la barre analysée. 

© 2000-2015, MetaQuotes Software Corp.

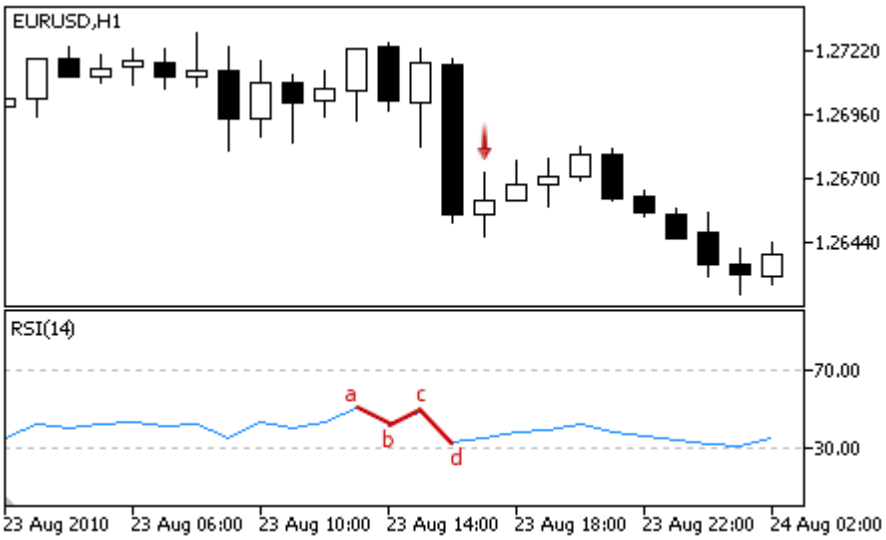
Type de Signal	Description des Conditions
	• **Divergence** — le premier creux analysé de l'oscillateur est plus bas que le précédent, et le creux correspondant des prix est plus bas que le précédent.  • **Double Divergence** — l'oscillateur forme 3 creux consécutifs, chacun d'eux est supérieur au précédent ; et les prix forment également les 3 creux correspondants, et chacun d'eux est plus bas que le précédent.  • **Tête/Epaules** — l'oscillateur forme trois creux consécutifs et celui du milieu est plus bas que les deux autres.

© 2000-2015, MetaQuotes Software Corp.

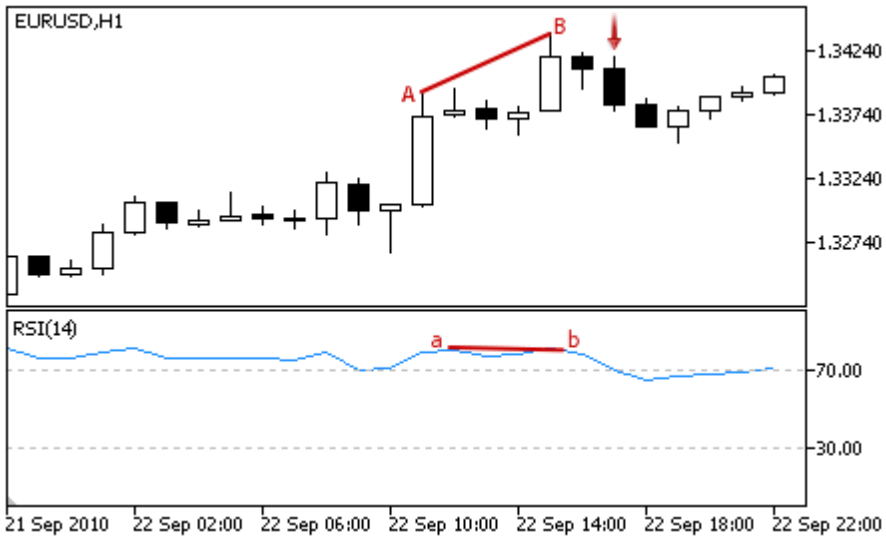
Type de Signal	Description des Conditions
	
For selling	• **Renversement en zone de sur-achat** — l'oscillateur se retourne à la baisse et sa valeur sur la barre en cours d'analyse est dans la zone de sur-achat (la valeur par défaut est 70).  • **Failed swing** — l'oscillator baisse plus bas que le creux précédent sur la barre analysée.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions

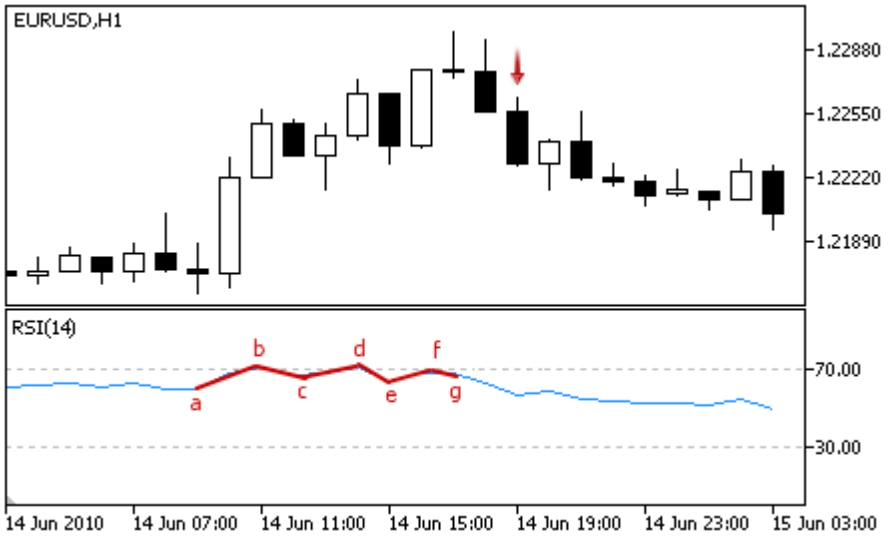


- **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent.



- **Double Divergence** — l'oscillateur forme 3 pics consécutifs, chacun d'eux est inférieur au précédent ; et les prix forment également les 3 pics correspondants, et chacun d'eux est plus haut que le précédent.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	 • **Tête/Epaules** — l'oscillateur forme 3 pics consécutifs et celui du milieu est plus haut que les 2 autres. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

© 2000-2015, MetaQuotes Software Corp.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodRSI	Période de calcul de l'oscillateur.
Applied	Une série de prix utilisée pour le calcul de l'oscillateur.

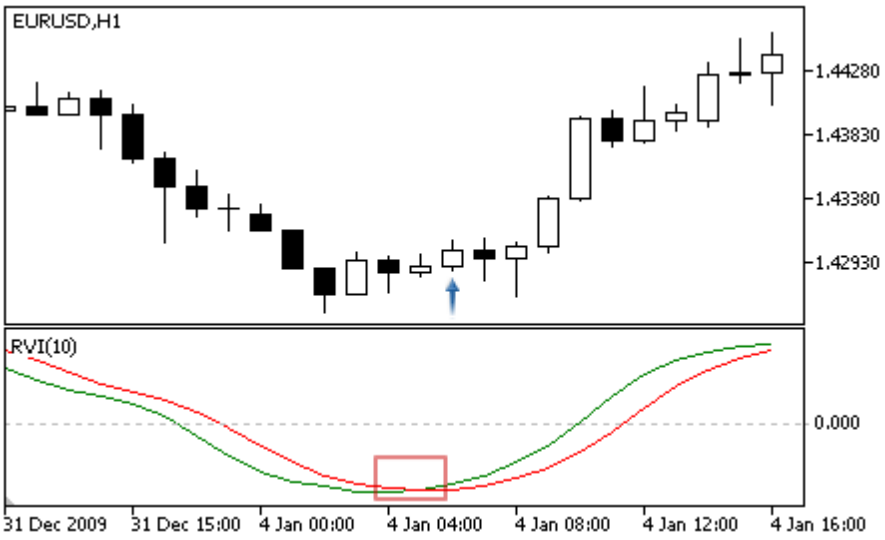
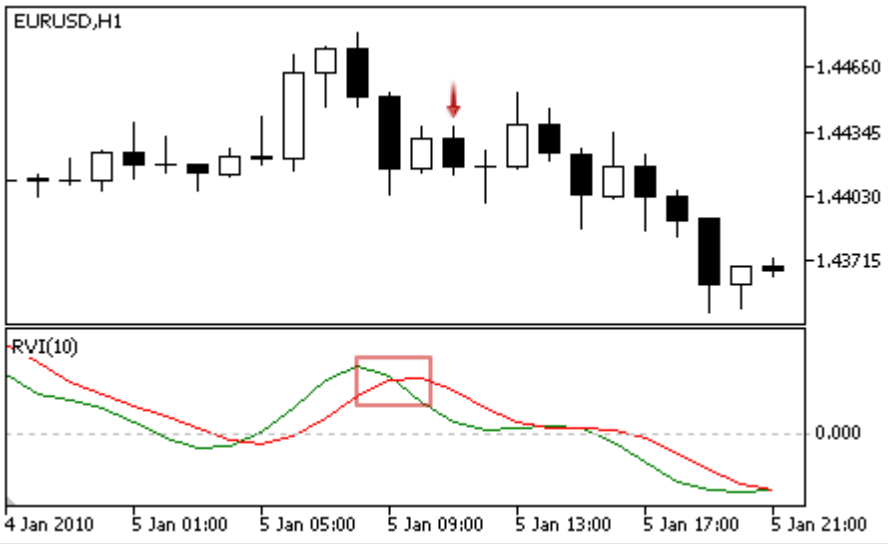
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'oscillateur Relative Vigor Index

Ce module de signaux est basé sur les modèles de marché de l'oscillateur Relative Vigor Index. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	**Croisement de la ligne principale et de la ligne de signal** — la ligne principale est au-dessus de la ligne du signal sur la barre analysée et sous la ligne de signal sur la barre précédente. 
For selling	**Croisement de la ligne principale et de la ligne de signal** — la ligne principale est en-dessous de la ligne du signal sur la barre analysée et au-dessus de la ligne de signal sur la barre précédente. 

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodRVI	Période de calcul de l'oscillateur.

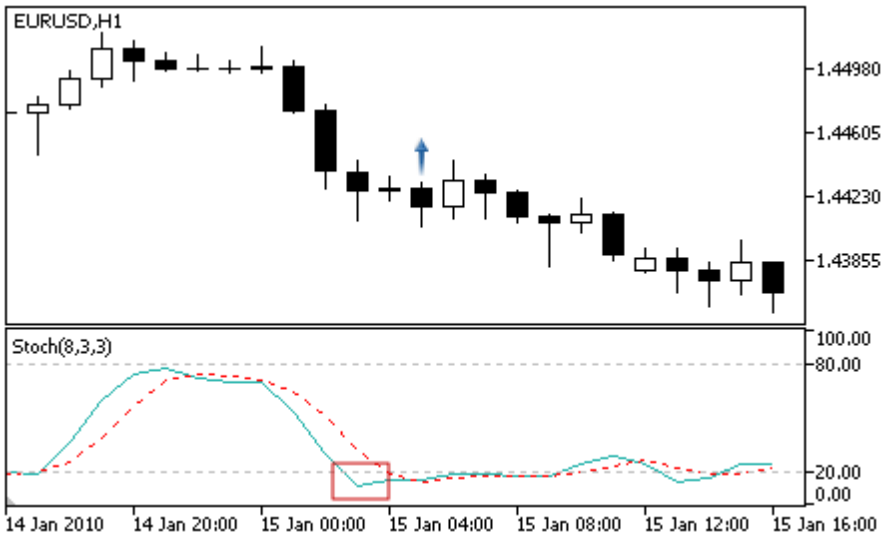
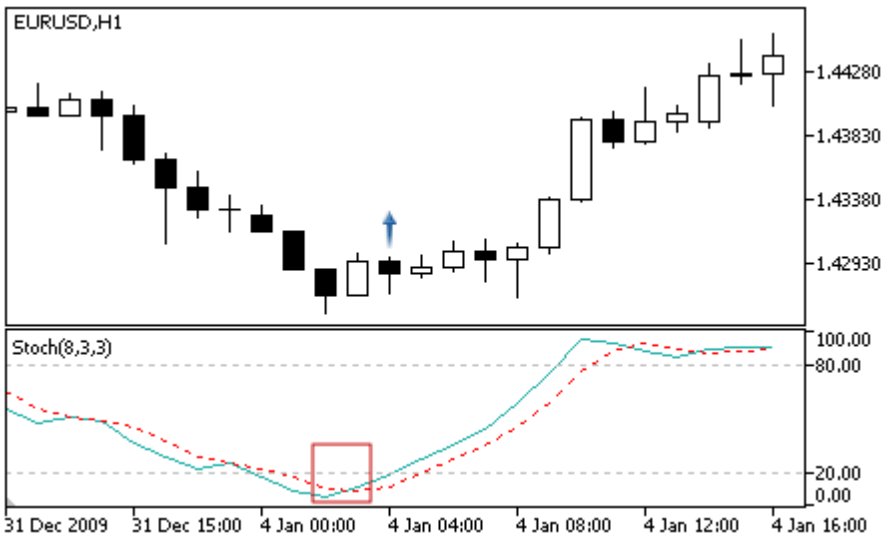
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'oscillateur Stochastic

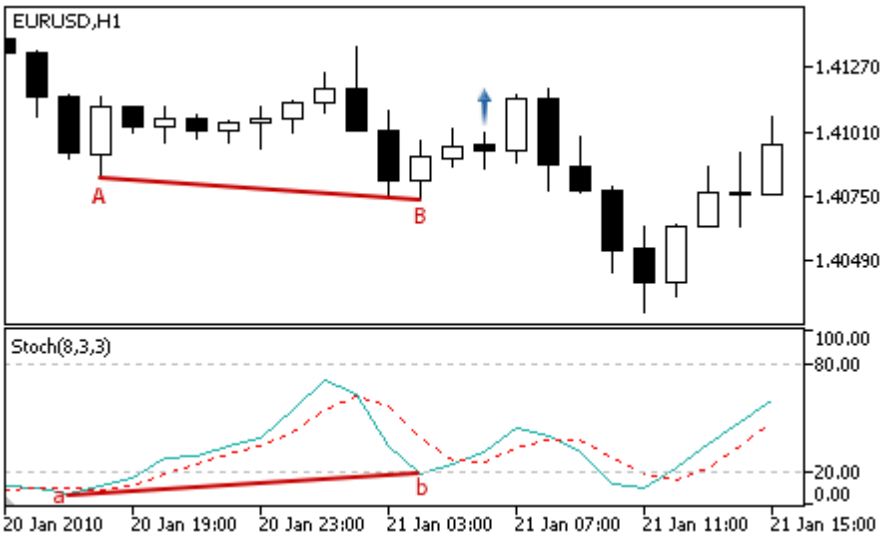
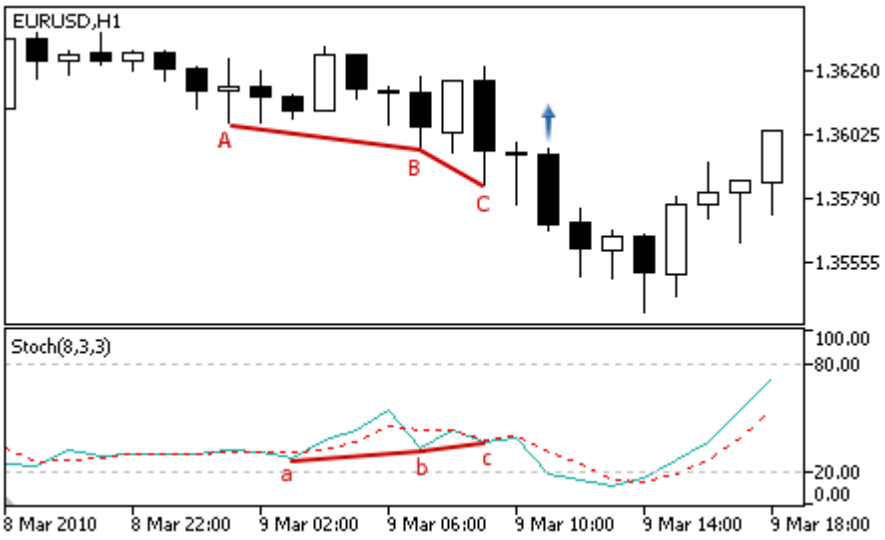
Ce module de signaux est basé sur les modèles de marché de l'oscillateur Stochastique. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

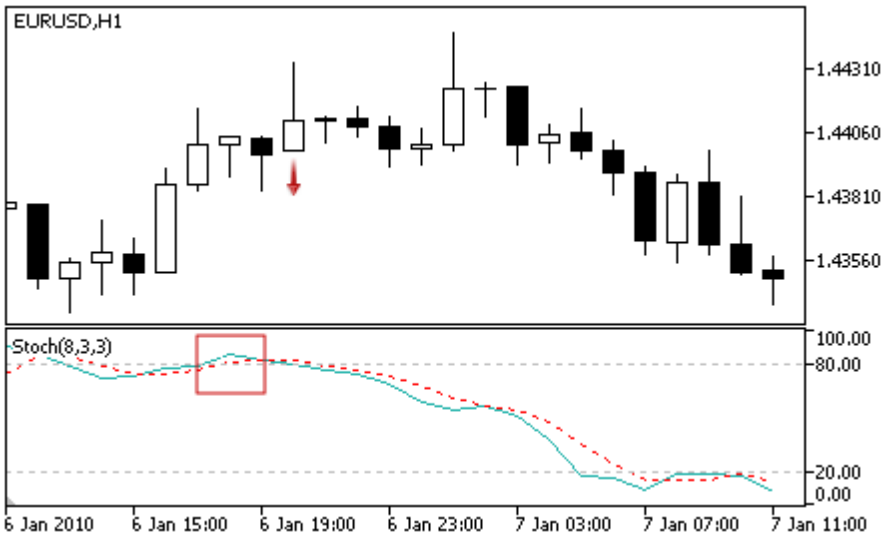
Type de Signal	Description des Conditions
Pour l'achat	• **Renversement** — l'oscillateur s'est retourné à la hausse (l'oscillateur monte sur la barre analysée et baisse sur la barre précédente).  • **Croisement de la ligne principale et de la ligne de signal** — la ligne principale est au-dessus de la ligne du signal sur la barre analysée et sous la ligne de signal sur la barre précédente. 

© 2000-2015, MetaQuotes Software Corp.

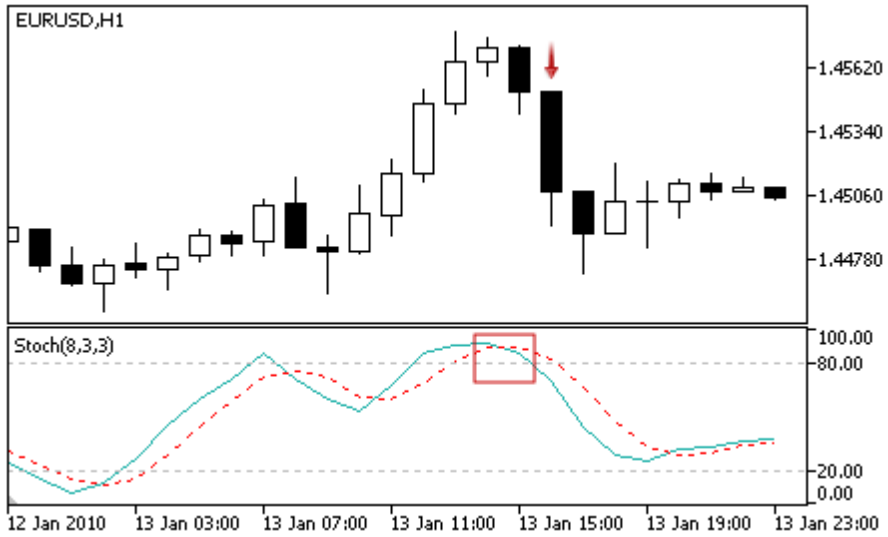
Type de Signal	Description des Conditions
	• **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent.  • **Double Divergence** — l'oscillateur forme 3 creux consécutifs, chacun d'eux est supérieur au précédent ; et les prix forment également les 3 creux correspondants, et chacun d'eux est plus bas que le précédent. 
For selling	• **Renversement** — l'oscillateur se retourne à la baisse (l'oscillateur est en baisse sur la barre analysée et en hausse sur la barre précédente).

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions

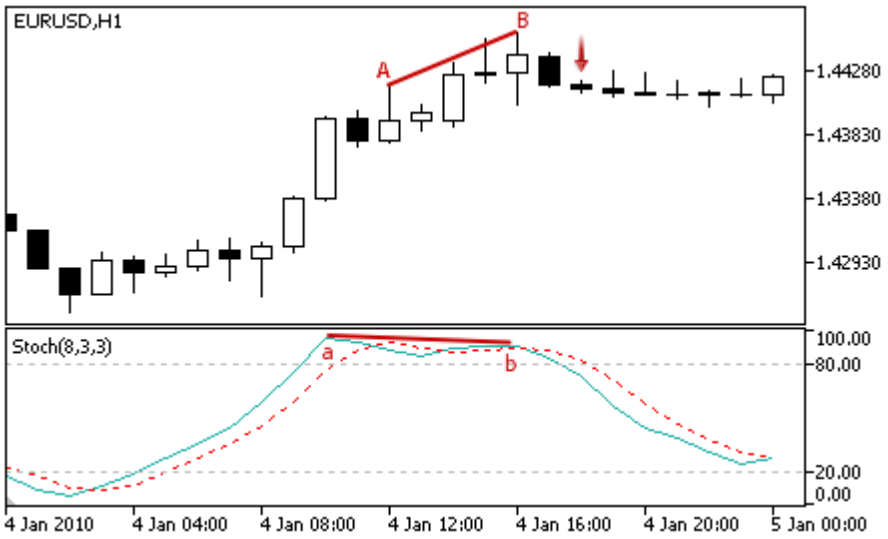
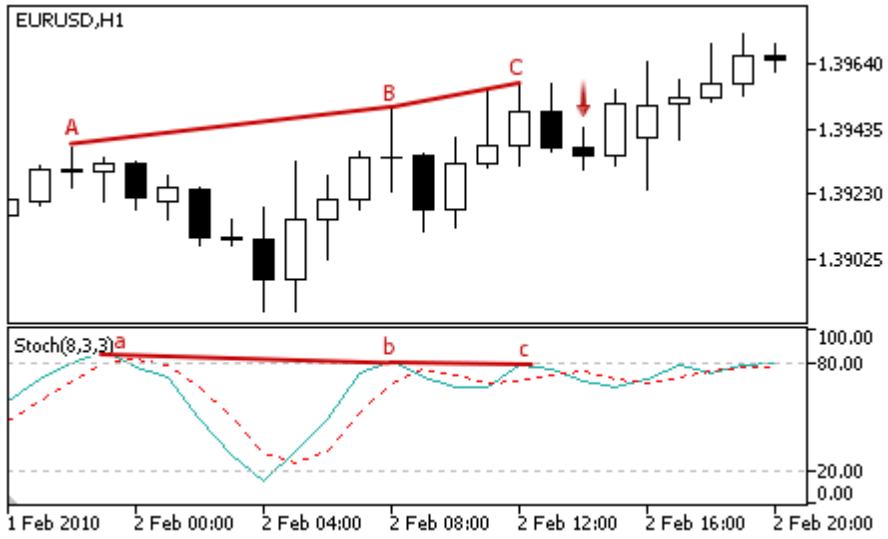


- **Croisement de la ligne principale et de la ligne de signal** — la ligne principale est en-dessous de la ligne du signal sur la barre analysée et au-dessus de la ligne de signal sur la barre précédente.



- **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	 • **Double Divergence** — l'oscillateur forme 3 pics consécutifs, chacun d'eux est inférieur au précédent ; et les prix forment également les 3 pics correspondants, et chacun d'eux est plus haut que le précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

© 2000-2015, MetaQuotes Software Corp.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodK	Période de calcul de la ligne de signal de l'oscillateur.
PeriodD	Période de calcul de la ligne de signal de l'oscillateur.
PeriodSlow	Période de lissage.
Applied	Une série de prix utilisée pour le calcul de l'oscillateur.

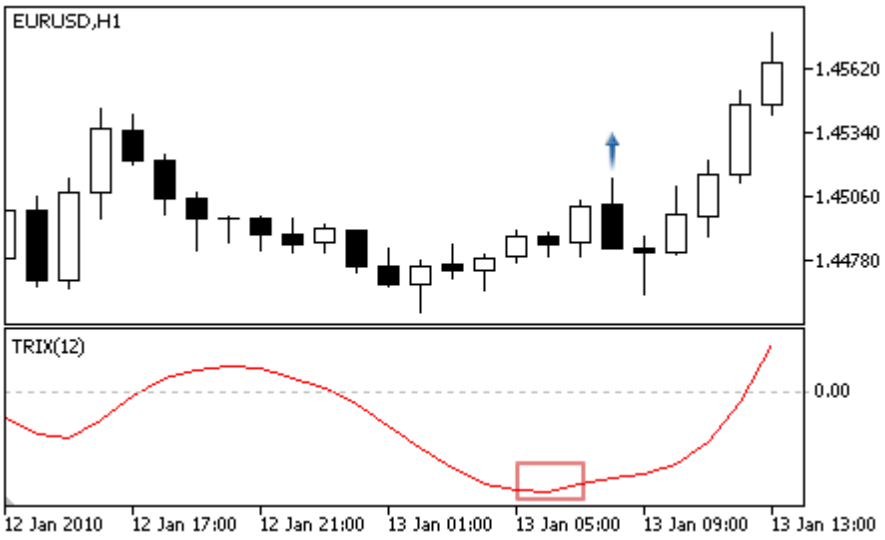
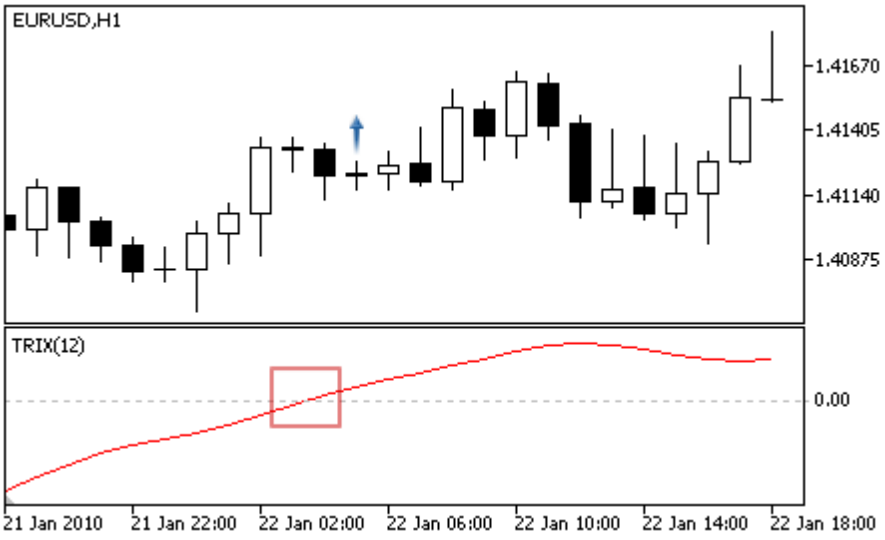
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'oscillateur Triple Exponential Average

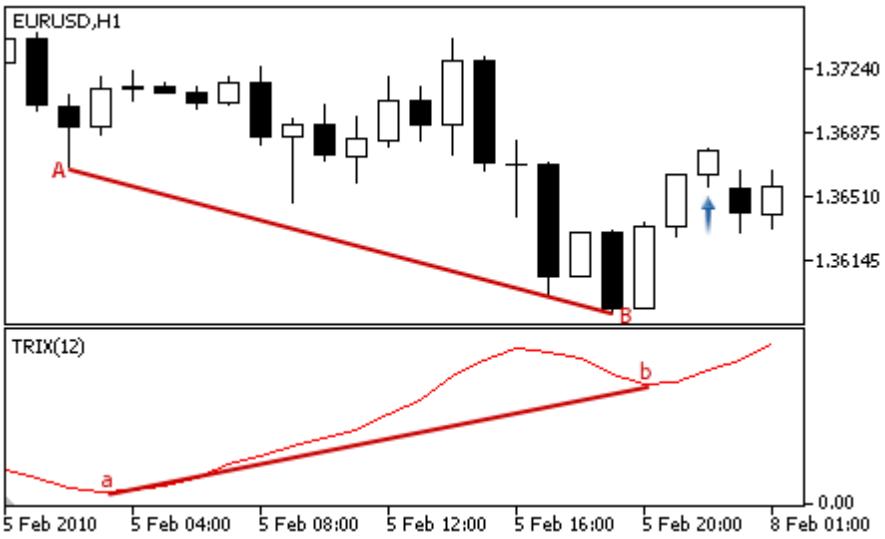
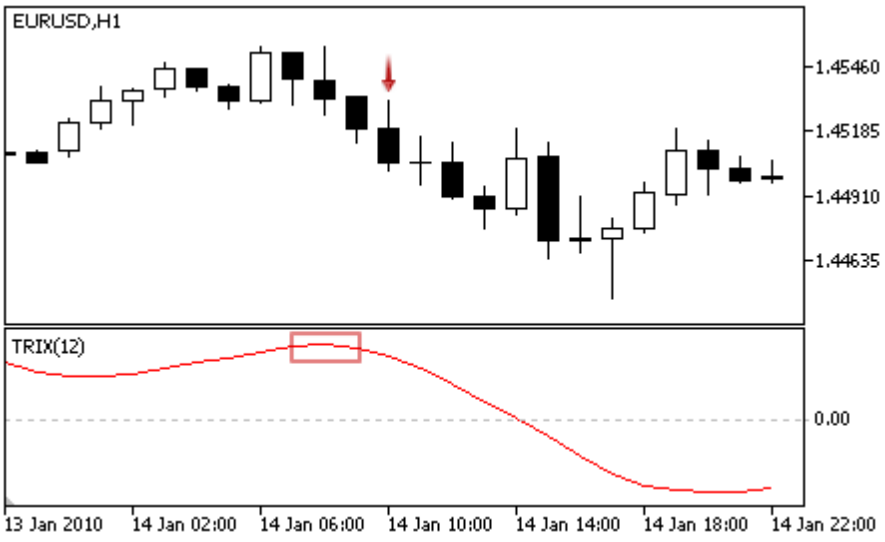
Ce module de signaux est basé sur les modèles de marché de l'indicateurTriple Exponential Average. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

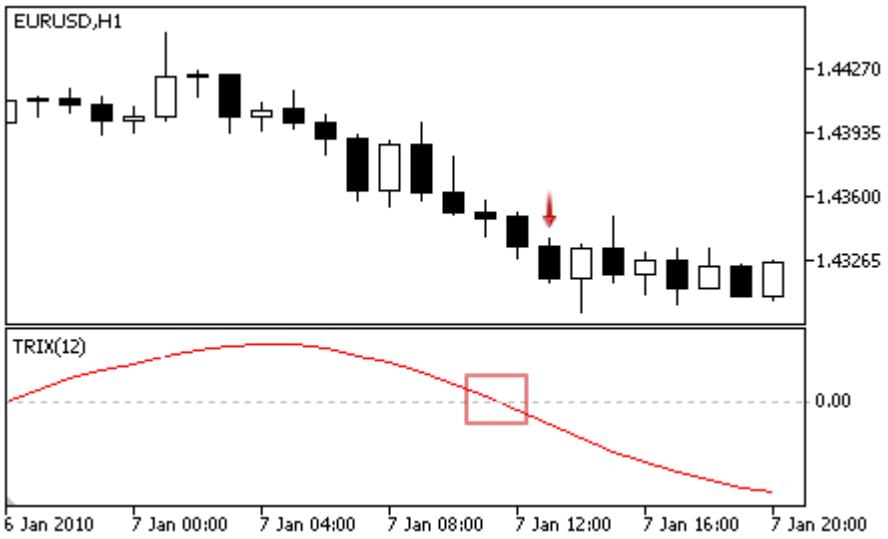
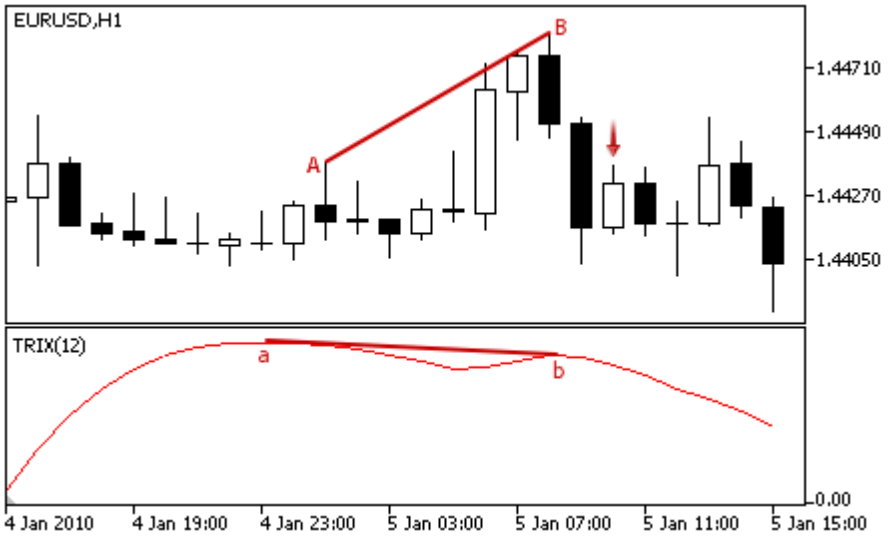
Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Renversement** — l'oscillateur s'est retourné à la hausse (l'oscillateur monte sur la barre analysée et baisse sur la barre précédente).  • **Croisement avec la ligne zéro** — la ligne de macd est au-dessus de zéro sur la barre analysée et sous la ligne de zéro sur la barre précédente. 

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	• **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent. 
For selling	• **Renversement** — l'oscillateur se retourne à la baisse (l'oscillateur est en baisse sur la barre analysée et en hausse sur la barre précédente).  • **Croisement avec la ligne zéro** — la ligne de macd est en-dessous de zéro sur la barre analysée et au-dessus de la ligne de zéro sur la barre précédente.

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	 • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

© 2000-2015, MetaQuotes Software Corp.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodTriX	Période de calcul de l'oscillateur.
Applied	Une série de prix utilisée pour le calcul de l'oscillateur.

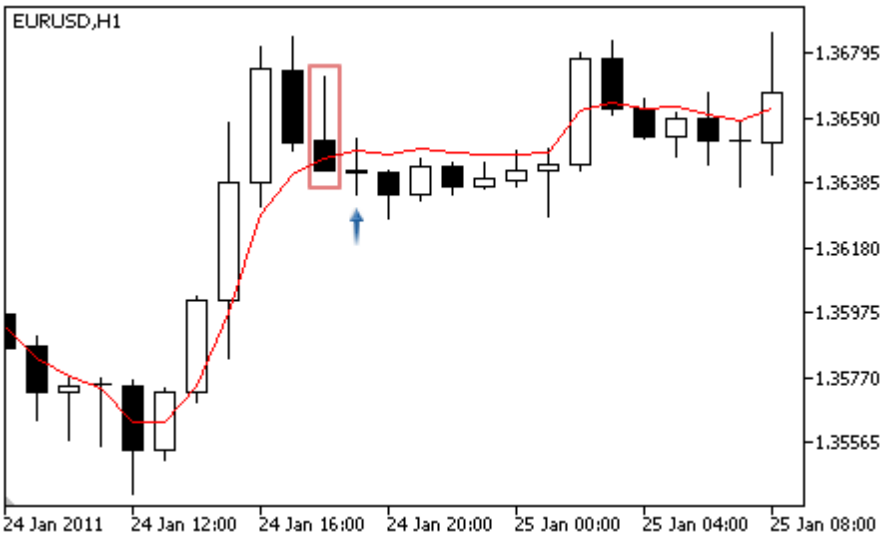
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'indicateur Triple Exponential Moving Average

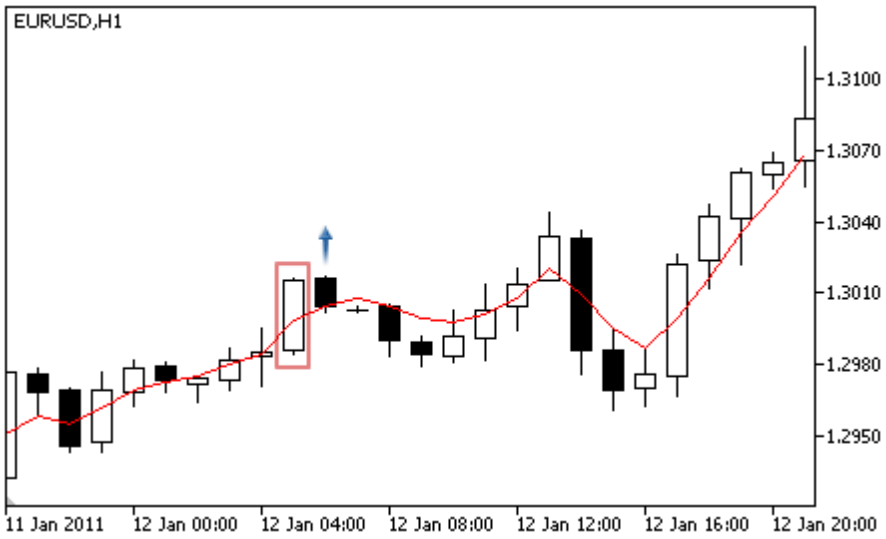
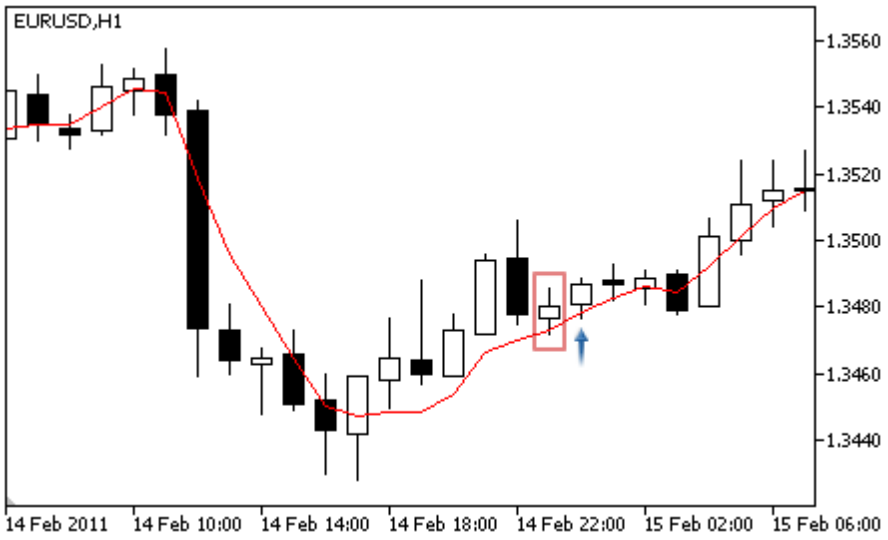
Ce module de signaux est basé sur les modèles de marché de l'indicateurTriple Exponential Moving Average. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

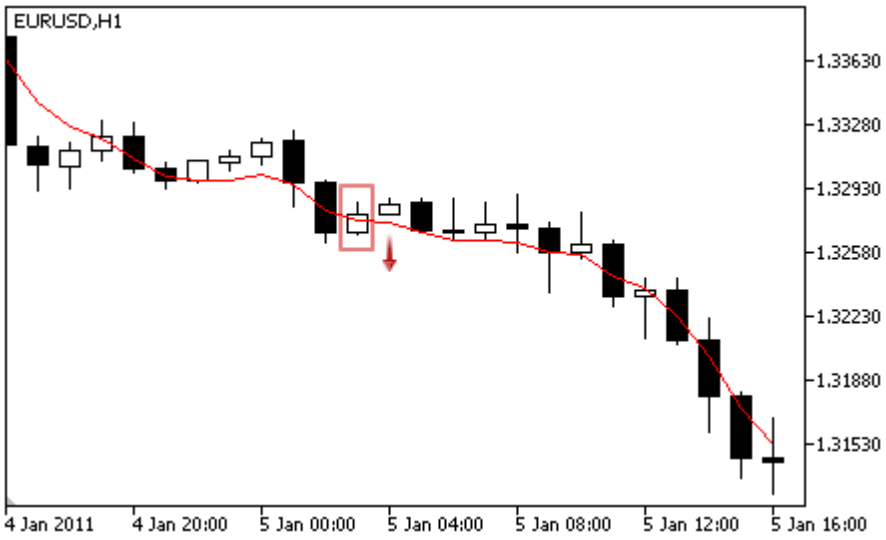
Type de Signal	Description des Conditions
Pour l'achat	• Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible).  • **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en hausse (signal fort).

© 2000-2015, MetaQuotes Software Corp.

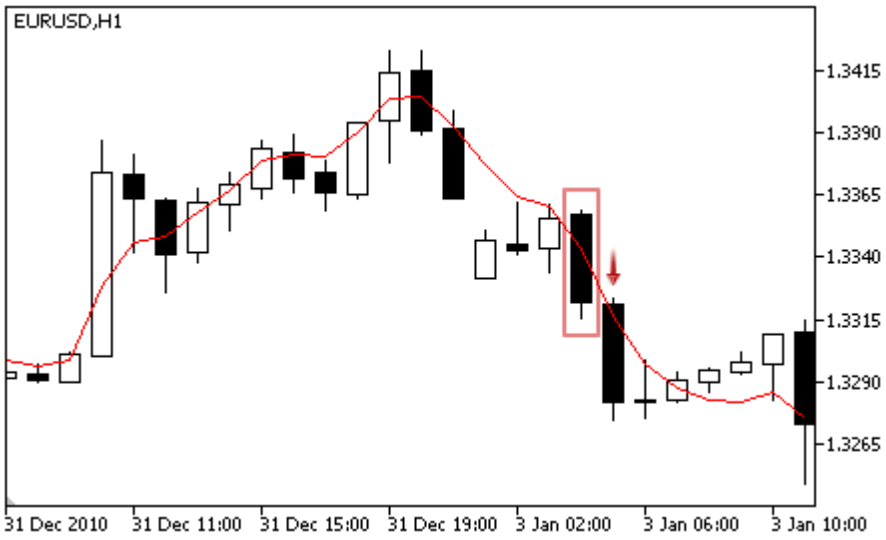
Type de Signal	Description des Conditions
	 • L'ombre inférieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont au-dessus de l'indicateur, et le prix le plus bas est en-dessous de l'indicateur) et l'indicateur est en hausse (signal faible). 
For selling	• Les prix ont croisé l'indicateur à la hausse (le prix d'ouverture de la barre en cours d'analyse est en-dessous de l'indicateur et le prix de clôture est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions

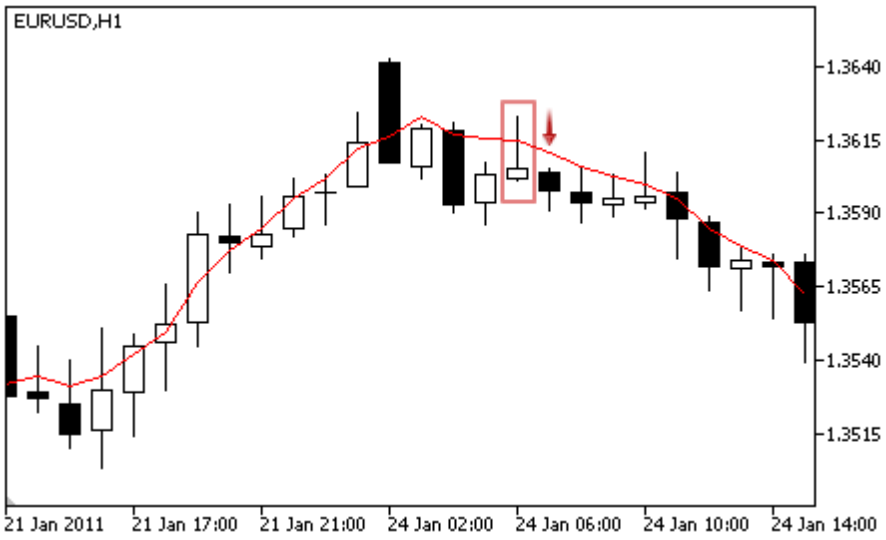


- **Croisement de la Moyenne Mobile.** Les prix ont croisé l'indicateur à la baisse (le prix d'ouverture de la barre en cours d'analyse est au-dessus de l'indicateur et le prix de clôture est en-dessous de l'indicateur) et l'indicateur est en baisse (signal fort).



- L'ombre supérieure de la barre a croisé l'indicateur (les prix d'ouverture et de clôture de la barre analysée sont en-dessous de l'indicateur, et le prix le plus haut est au-dessus de l'indicateur) et l'indicateur est en baisse (signal faible).

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
	
Aucune objection pour l'achat	Les prix sont au-dessus de l'indicateur.
Aucune objection pour la vente	Les prix sont en-dessous de l'indicateur.

Note

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodMA	Période de lissage de l'indicateur.
Shift	Décalage de l'indicateur le long de l'axe du temps (en barres).
Method	Méthode de lissage.
Applied	Une série de prix utilisée pour le calcul de l'indicateur.

© 2000-2015, MetaQuotes Software Corp.

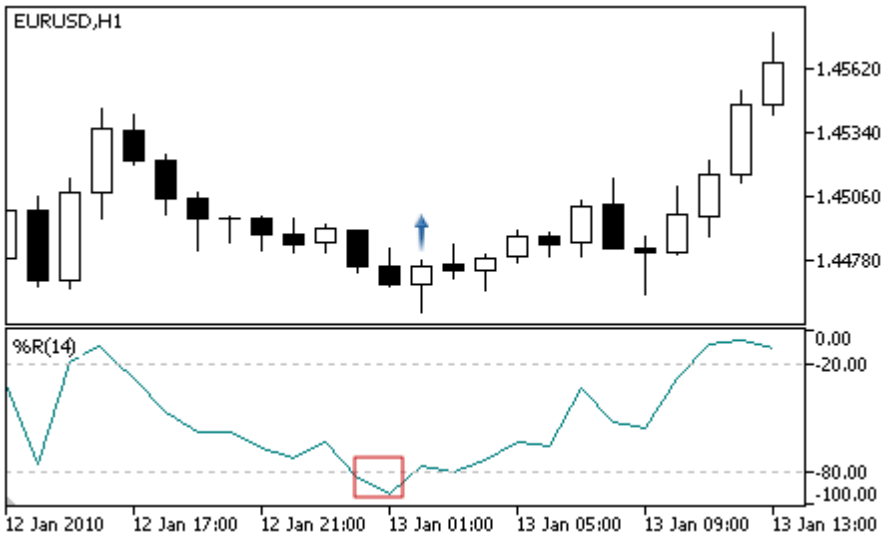
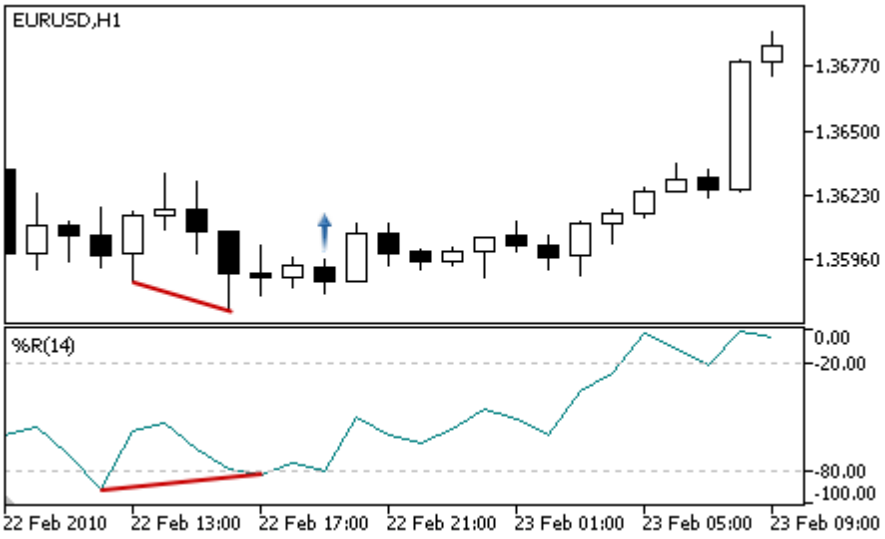
© 2000-2015, MetaQuotes Software Corp.

Signaux de l'oscillatuer Williams Percent Range

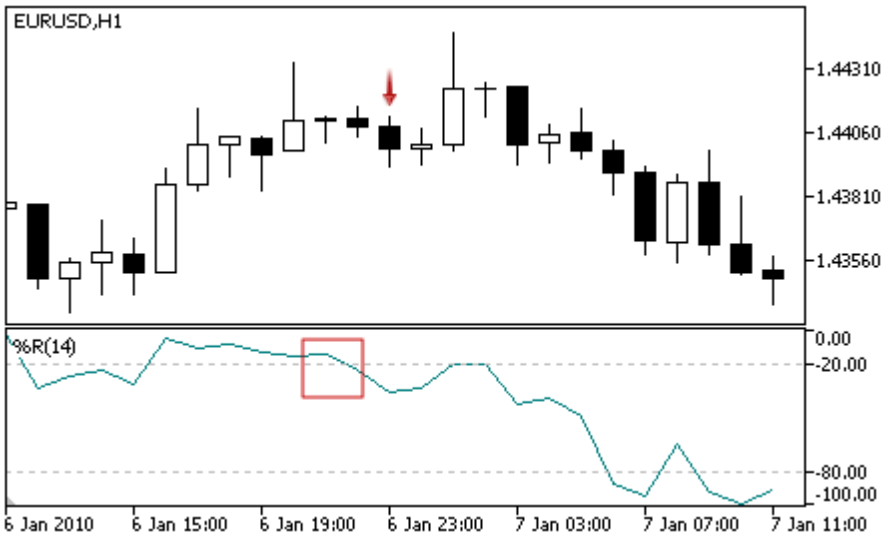
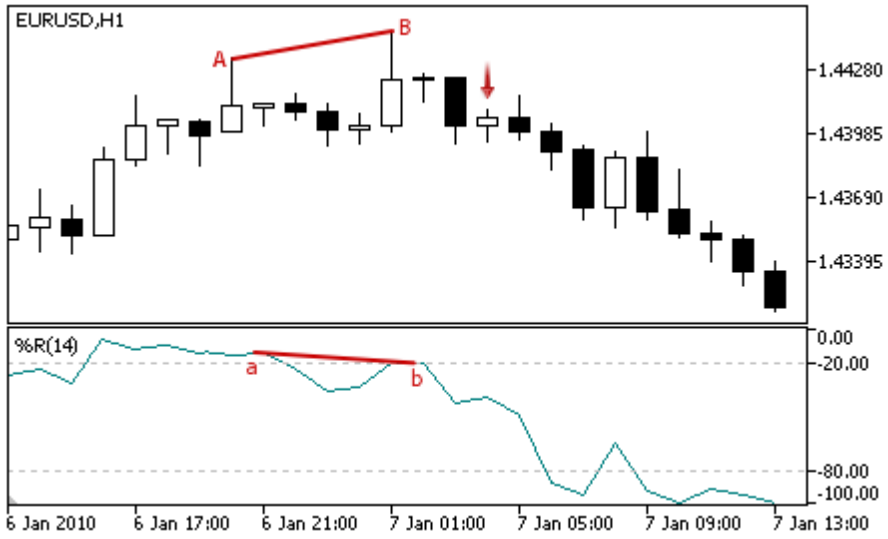
Ce module de signaux est basé sur les modèles de marché de l'oscillateur Williams Percent Range. Le mécanisme de prise de décisions basées sur les signaux et obtenu des modules est décrit dans une section dédiée.

Conditions de Génération des Signaux

Vous trouverez ci-dessous la description des conditions lorsque le module envoie un signal à un Expert Advisor.

Type de Signal	Description des Conditions
Pour l'achat	• **Retournement dans la zone de sur-vente** — l'oscillateur se retourne à la hausse et sa valeur sur la barre analysée est dans la zone de sur-vente (la valeur par défaut est -20).  • **Divergence** — le premier creux de l'oscillateur est plus haut que le précédent, et le creux correspondant des prix est plus bas que le précédent. 

© 2000-2015, MetaQuotes Software Corp.

Type de Signal	Description des Conditions
For selling	• **Renversement en zone de sur-achat** — l'oscillateur se retourne à la baisse et sa valeur sur la barre en cours d'analyse est dans la zone de sur-achat (la valeur par défaut est -80).  • **Divergence** — le premier pic analysé de l'oscillateur est plus bas que le précédent, et le pic correspondant des prix est plus haut que le pic précédent. 
Aucune objection pour l'achat	La valeur de l'indicateur est en hausse sur la barre analysée.
Aucune objection pour la vente	La valeur de l'indicateur baisse sur la barre analysée.

Note

© 2000-2015, MetaQuotes Software Corp.

Suivant le mode d'éxécution d'un Expert Advisor ("Every tick" ou "Open prices only"), une barre en cours d'analyse est soit la barre courante (avec index 0), ou la dernière barre formée (avec index 1).

Noter que l'oscillateur Williams Percent Range a une échelle inversée. Sa valeur maximum est -100, et la minimum est 0.

Paramètres Modifiables

Ce module a les paramètres modifiables suivants :

Paramètre	Description
Weight	Poids du signal du module dans l'intervalle 0 à 1.
PeriodWPR	Période de calcul de l'oscillateur.

© 2000-2015, MetaQuotes Software Corp.

Classes de Trailing Stop (Stops Suiveurs)

Cette section contient les détails techniques d'utilisation des classes de stops suiveurs et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création (et du test) de vos stratégies de trading.

La Bibliothèque Standard MQL5 (en terme de stratégies de trading) est située dans le répertoire de travail du terminal dans le répertoire Include\Expert\Trailing.

Classe	Description
CTrailingFixedPips	Cette classe implémente un algorithme de stop suiveur, basé sur des points fixes
CTrailingMA	Cette classe implémente un algorithme de stop suiveur, basé sur les valeurs d'une moyenne mobile
CTrailingNone	Classe de type stub, elle n'utilise aucun algorithme de Stop Suiveur
CTrailingPSAR	Cette classe implémente un algorithme de stop suiveur, basé sur les valeurs de l'indicateur Parabolic SAR

© 2000-2015, MetaQuotes Software Corp.

CTrailingFixedPips

La classe CTrailingFixedPips implémente un algorithme de stop suiveur, basé sur un suivi avec des points fixes.

Si la position a un Stop Loss, vérifie la distance minimale autorisée du Stop Loss au prix courant. Si sa valeur est inférieure au prix du Stop Loss, cela suggère de définir un nouveau prix pour le Stop Loss. Si la position a un Take Profit, suggère de définir un nouveau prix pour le Take Profit.

Si l'Expert Advisor a été initialisé avec le flag every_tick=faux, la class n'effectuera toutes les opérations (trading, suivi, etc.) que sur la nouvelle barre. Dans ce cas, le niveau du Take Profit peut être utilisé. Cela vous permettra de clôturer la position ouverte au prix du Take Profit avant que la nouvelle barre ne soit complétée.

Description

CTrailingFixedPips implémente l'algorithme du Stop Suiveur basé sur une taille fixe par rapport aux positions.

Déclaration

```
class CTrailingFixedPips: public CExpertTrailing
```

Titre

```
   #include <Expert\Trailing\CTrailingFixedPips.mqh>
```

Méthodes de Classe

Initialisation	
StopLevel	Définit la valeur du niveau du Stop Loss
ProfitLevel	Définit la valeur du niveau du Take Profit
virtual ValidationSettings	Vérifie les paramètres
Méthodes de Vérifications des Suivis	
virtual CheckTrailingStopLong	Vérifie les conditions du Stop Suiveur d'une position longue
virtual CheckTrailingStopShort	Vérifie les conditions du Stop Suiveur d'une position courte

© 2000-2015, MetaQuotes Software Corp.

StopLevel

Définit la valeur du niveau du Stop Loss (en points).

```
void  StopLevel(
   int  stop_level      // Stop Loss
   )
```

Paramètres

stop_loss

[in] La Nouvelle valeur du niveau du Stop Loss (selon la convention des points à 2/4 décimales).

Note

Si le niveau du Stop Loss est égal à 0, le Stop Suiveur n'est pas utilisé.

© 2000-2015, MetaQuotes Software Corp.

ProfitLevel

Définit la valeur du niveau du Take Profit (en points).

```
void  ProfitLevel(
   int  profit_level      // Take profit
   )
```

Paramètres

profit_level

[in] Nouvelle valeur du niveau du Take Profit (selon la convention des points à 2/4 décimales).

Note

Si le niveau du Take Profit est égal à 0, le Stop Suiveur n'est pas utilisé.

© 2000-2015, MetaQuotes Software Corp.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La fonction vérigie les niveaux du Take Profit et du Stop Loss. Les valeurs correctes sont 0 et les valeurs supérieures au stop minimal des ordres stops du symbole.

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopLong

Vérifie les conditions du Stop Suiveur d'une position longue.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // pointeur sur un objet CPositionInfo
   double&         sl,           // Prix du Stop Loss
   double&         tp            // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Si le niveau du Stop Loss est égal à 0, le Stop Suiveur n'est pas utilisé. Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est considérée comme le prix de base.

Si le prix Bid (prix de l'offre) est supérieur au prix de base+stop loss, cela suggère de définir un nouveau prix de Stop Loss. Dans ce cas, si la position a déjà un Take Profit, cela suggère de définir un nouveau Take Profit égal au niveau prix Bid +Take Profit.

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopShort

Vérifie les conditions du Stop Suiveur d'une position courte.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // pointeur sur un objet CPositionInfo
   double&         sl,           // Prix du Stop Loss
   double&         tp            // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Si le niveau du Stop Loss est égal à 0, le Stop Suiveur n'est pas utilisé. Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est considérée comme le prix de base.

If the current Ask price is lower than base price-stop loss level, it suggests to set new Prix du Stop Loss Dans ce cas, si la position a déjà un Take Profit, cela suggère de définir un nouveau Take Profit égal au niveau prix Ask - Take Profit.

© 2000-2015, MetaQuotes Software Corp.

CTrailingMA

La classe CTrailingMA implémente un algorithme de stop suiveur, basé sur les valeurs d'une moyenne mobile.

Description

La classe CTrailingMA implémente un algorithme de stop suiveur, basé sur les valeurs d'une moyenne mobile

Déclaration

```
class CTrailingMA: public CExpertTrailing
```

Titre

```
   #include <Expert\Trailing\TrailingMA.mqh>
```

Méthodes de Classe

Initialisation	
Period	Définit la période de la moyenne mobile.
Shift	Définit le décalage de la moyenne mobile.
Method	Définit la méthode de lissage de la moyenne mobile.
Applied	Définit le prix à appliquer à la moyenne mobile.
virtual InitIndicators	Initialise les indicateurs et les séries de données.
virtual ValidationSettings	Vérifie les paramètres
Méthodes de Vérifications des Suivis	
virtual CheckTrailingStopLong	Vérifie les conditions du Stop Suiveur d'une position longue
virtual CheckTrailingStopShort	Vérifie les conditions du Stop Suiveur d'une position courte

© 2000-2015, MetaQuotes Software Corp.

Period

Définit la période de la moyenne mobile.

```
void  Period(
   int  period      // Période de lissage
   )
```

Paramètres

period

[in] Période de la moyenne mobile.

© 2000-2015, MetaQuotes Software Corp.

Shift

Définit le décalage de la moyenne mobile.

```
void  Shift(
   int  shift      // Décalage
   )
```

Paramètres

shift

[in] Décalage de la moyenne mobile.

© 2000-2015, MetaQuotes Software Corp.

Method

Définit la méthode de lissage de la moyenne mobile.

```
void  Method(
   ENUM_MA_METHOD  method      // Méthode de lissage
   )
```

Paramètres

method

[in] Méthode de lissage de la moyenne mobile.

© 2000-2015, MetaQuotes Software Corp.

Applied

Définit le prix à appliquer à la moyenne mobile.

```
void  Applied(
   ENUM_APPLIED_PRICE  applied      // Prix appliqué
   )
```

Paramètres

applied

[in] Prix appliqué à la moyenne mobile.

© 2000-2015, MetaQuotes Software Corp.

InitIndicators

Initialise les indicateurs et les séries de données.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // Pointeur vers une collection de CIndicators
   )
```

Paramètres

indicators

[in] Pointeur vers une collection d'indicateurs et de séries de données (membre de la classe CExpert).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La fonction vérifie la période de la moyenne mobile, les valeurs correctes sont positives.

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopLong

Vérifie les conditions du Stop Suiveur d'une position longue.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // pointeur sur un objet CPositionInfo
   double&         sl,           // Prix du Stop Loss
   double&         tp            // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Le prix maximum du Stop Loss est d'abord calculé, le plus près du prix courant en utilisant les valeurs de la moyenne mobile de la barre précédente (barre complète).

Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est utilisé.

Si le prix calculé du Stop Loss est supérieur au prix de base et inférieur au prix maximum du Stop Loss autorisé, cela suggère de définir un nouveau prix de Stop Loss.

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopShort

Vérifie les conditions du Stop Suiveur d'une position courte.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // pointeur sur un objet CPositionInfo
   double&         sl,           // Prix du Stop Loss
   double&         tp            // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Le prix maximum du Stop Loss est d'abord calculé, le plus près du prix courant en utilisant les valeurs de la moyenne mobile de la barre précédente (barre complète).

Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est utilisé.

Si le prix calculé du Stop Loss est supérieur au prix de base et inférieur au prix minimum du Stop Loss autorisé, cela suggère de définir un nouveau prix de Stop Loss.

© 2000-2015, MetaQuotes Software Corp.

CTrailingNone

La classe CTrailingNone est un stub. Cette classe devrait être utilisée au moment de l'initialisation de l'objet de Suivi si votre stratégie n'utilise pas de Stop Suiveur.

Description

La classe CTrailingNone n'implémente aucun algorithme de Stop Suiveur. Les méthodes de vérification des conditions des Stops Suiveurs retournent toujours faux.

Déclaration

```
class CTrailingNone: public CExpertTrailing
```

Titre

```
   #include <Expert\Trailing\TrailingNone.mqh>
```

Méthodes de Classe

Méthodes de Vérifications des Suivis	
virtual CheckTrailingStopLong	Une méthode stub pour vérifier les conditions du Stop Suiveur d'une position longue
virtual CheckTrailingStopShort	Une méthode stub pour vérifier les conditions du Stop Suiveur d'une position courte

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopLong

Vérifie les conditions du Stop Suiveur d'une position longue.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // pointeur sur un objet CPositionInfo
   double&         sl,           // Prix du Stop Loss
   double&         tp            // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

La fonction retourne toujours faux.

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopShort

Vérifie les conditions du Stop Suiveur d'une position courte.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // pointeur sur un objet CPositionInfo
   double&         sl,           // Prix du Stop Loss
   double&         tp            // Prix du Take Profit
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

La fonction retourne toujours faux.

© 2000-2015, MetaQuotes Software Corp.

CTrailingPSAR

CTrailingPSAR est une classe d'implémentation de l'algorithme de Stop Suiveur basé sur les valeurs de l'indicateur Parabolic SAR.

Description

La classe CTrailingPSAR implémente l'algorithme de Stop Suiveur basé sur les valeurs de l'indicateur Parabolic SAR sur la barre précédente (complétée).

Déclaration

```
class CTrailingPSAR: public CExpertTrailing
```

Titre

```
   #include <Expert\Trailing\TrailingParabolicSAR.mqh>
```

Méthodes de Classe

Initialisation	
Step	Définit la valeur de pas de l'indicateur Parabolic SAR
Maximum	Définit la valeur maximum de l'indicateur Parabolic SAR
virtual InitIndicators	Initialise les indicateurs et les séries de données.
Méthodes de Vérifications des Suivis	
virtual CheckTrailingStopLong	Vérifie les conditions du Stop Suiveur d'une position longue
virtual CheckTrailingStopShort	Vérifie les conditions du Stop Suiveur d'une position courte

© 2000-2015, MetaQuotes Software Corp.

Step

Définit la valeur de pas de l'indicateur Parabolic SAR.

```
void  Step(
   double  step      // Pas
   )
```

Paramètres

step

[in] La valeur du pas de l'indicateur Parabolic SAR.

© 2000-2015, MetaQuotes Software Corp.

Maximum

Définit la valeur maximum de l'indicateur Parabolic SAR.

```
void  Maximum(
   double  maximum      // Maximum
   )
```

Paramètres

maximum

[in] La valeur du maximum de l'indicateur Parabolic SAR.

© 2000-2015, MetaQuotes Software Corp.

InitIndicators

Initialise les indicateurs et les séries de données.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // Pointeur vers une collection de CIndicators
   )
```

Paramètres

indicators

[in] Pointeur vers une collection d'indicateurs et de séries de données (membre de la classe CExpert).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopLong

Vérifie les conditions du Stop Suiveur d'une position longue.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // pointeur
   double&         sl,           // Lien
   double&         tp            // Lien
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Le prix maximum du Stop Loss est d'abord calculé, le plus près du prix courant en utilisant les valeurs de l'indicateur Parabolic SAR de la barre précédente (barre complète).

Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est considérée comme le prix de base.

Si le prix calculé du Stop Loss est supérieur au prix de base et inférieur au prix maximum du Stop Loss autorisé, cela suggère de définir un nouveau prix de Stop Loss.

© 2000-2015, MetaQuotes Software Corp.

CheckTrailingStopShort

Vérifie les conditions du Stop Suiveur d'une position courte.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // pointeur
   double&         sl,           // Lien
   double&         tp            // Lien
   )
```

Paramètres

position

[in] Pointeur sur un objet CPositionInfo.

sl

[in][out] Variable du prix du Stop Loss

tp

[in][out] Variable for Prix du Take Profit

Valeur de retour

vrai si la condition est satisfaite, faux sinon.

Note

Le prix maximum du Stop Loss est d'abord calculé, le plus près du prix courant en utilisant les valeurs de l'indicateur Parabolic SAR de la barre précédente (barre complète).

Si la position a déjà un Stop Loss, sa valeur est considérée comme étant le prix de base, sinon le prix d'ouverture de la position est considérée comme le prix de base.

Si le prix calculé du Stop Loss est supérieur au prix de base et inférieur au prix minimum du Stop Loss autorisé, cela suggère de définir un nouveau prix de Stop Loss.

© 2000-2015, MetaQuotes Software Corp.

Classes de Money Management

Cette section contient les détails techniques d'utilisation des classes de gestion du capital et du risque (money et risk management) et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création (et du test) de vos stratégies de trading.

La Bibliothèque Standard MQL5 (en terme de classes de money management et de risk management) est située dans le répertoire de travail du terminal dans le répertoire Include\Expert\Money\.

Classe	Description
CMoneyFixedLot	Cette classe implémente un algorithme de money management basé sur le trading avec une taille de lot fixe et prédéfinie.
CMoneyFixedMargin	Cette classe implémente un algorithme de money management basé sur le trading avec une marge fixe et prédéfinie.
CMoneyFixedRisk	Cette classe implémente un algorithme de money management basé sur le trading avec un risque prédéfini.
CMoneyNone	Cette classe implémente un algorithme de money management basé sur le trading une taille de lot minimum autorisée.
CMoneySizeOptimized	Cette classe implémente un algorithme de money management basé sur le trading avec une taille de lot variable, en fonction des résultats des deals précédents.

© 2000-2015, MetaQuotes Software Corp.

CMoneyFixedLot

CMoneyFixedLot est la classe de l'algorithme de money management, basé sur le trading avec une taille de lot fixe.

Description

CMoneyFixedLot implémente l'algorithme de money management, basé sur le trading avec une taille de lot fixe.

Déclaration

```
class CMoneyFixedLot: public CExpertMoney
```

Titre

```
#include <Expert\Money\MoneyFixedLot.mqh>
```

Méthodes de Classe

Initialisation	
Lots	Définit le volume de trading
virtual ValidationSettings	Vérifie les paramètres
Méthodes de Money et de Risk Management (Gestion du Capital et du Risque)	
virtual CheckOpenLong	Retourne le volume de trading pour une position longue
virtual CheckOpenShort	Retourne le volume de trading pour une position courte

© 2000-2015, MetaQuotes Software Corp.

Lots

Définit le volume de trading (en lots).

```
void  Lots(
   double  lots      // Lots
   )
```

Paramètres

lots

[in] Volume de trading (en lots).

© 2000-2015, MetaQuotes Software Corp.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Vérifie le volume de trading spécifié.

© 2000-2015, MetaQuotes Software Corp.

CheckOpenLong

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenLong(
   double  price,     // prix
   double  sl         // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne toujours le volume de trading fixé, défini par la méthode Lots.

© 2000-2015, MetaQuotes Software Corp.

CheckOpenShort

Retourne le volume de trading pour une position courte.

```
virtual double  CheckOpenShort(
   double  price,     // prix
   double  sl         // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position courte.

Note

La fonction retourne toujours le volume de trading fixé, défini par la méthode Lots.

© 2000-2015, MetaQuotes Software Corp.

CMoneyFixedMargin

CMoneyFixedMargin est la classe de l'algorithme de money management, basé sur le trading avec une marge fixe prédéfinie.

Description

CMoneyFixedMargin implément l'algorithme de money management, basé sur le trading avec une marge fixe prédéfinie.

Déclaration

```
class CMoneyFixedMargin: public CExpertMoney
```

Titre

```
#include <Expert\Money\MoneyFixedMargin.mqh>
```

Méthodes de Classe

Méthodes de Money et de Risk Management (Gestion du Capital et du Risque)	
virtual CheckOpenLong	Retourne le volume de trading pour une position longue
virtual CheckOpenShort	Retourne le volume de trading pour une position courte

© 2000-2015, MetaQuotes Software Corp.

CheckOpenLong

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenLong(
   double  price,     // prix
   double  sl         // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne le volume de trading pour une position longue, elle utilise la marge fixe. La marge est défini par le paramètre "Percent" de la classe de base CExpertMoney.

© 2000-2015, MetaQuotes Software Corp.

CheckOpenShort

Retourne le voume de trading pour une position courte.

```
virtual double  CheckOpenShort(
   double  price,     // prix
   double  sl         // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position courte.

Note

La fonction retourne le volume de trading pour une position courte, elle utilise la marge fixe. La marge est défini par le paramètre "Percent" de la classe de base CExpertMoney.

© 2000-2015, MetaQuotes Software Corp.

CMoneyFixedRisk

CMoneyFixedRisk est la classe pour implémenter un algorithme de money management avec un risque fixe et prédéfini.

Description

La classe CMoneyFixedRisk implémente l'algorithme de money management avec un risque fixe et prédéfini.

Déclaration

```
class CMoneyFixedRisk: public CExpertMoney
```

Titre

```
#include <Expert\Money\MoneyFixedRisk.mqh>
```

Méthodes de Classe

Méthodes de Money et de Risk Management (Gestion du Capital et du Risque)	
virtual CheckOpenLong	Retourne le volume de trading pour une position longue
virtual CheckOpenShort	Retourne le volume de trading pour une position courte

© 2000-2015, MetaQuotes Software Corp.

CheckOpenLong

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenLong(
   double  price,     // prix
   double  sl         // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne le volume de trading pour une position longue, elle utilise un risque fixe. Le risque est défini par le paramètre "Percent" de la classe de base CExpertMoney.

© 2000-2015, MetaQuotes Software Corp.

CheckOpenShort

Retourne le volume de trading pour une position courte.

```
virtual double  CheckOpenShort(
   double  price,     // prix
   double  sl         // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position courte.

Note

La fonction retourne le volume de trading pour une position courte, elle utilise un risque fixe. Le risque est défini par le paramètre "Percent" de la classe de base CExpertMoney.

© 2000-2015, MetaQuotes Software Corp.

CMoneyNone

CMoneyNone est une classe implémentant l'algorithme de trading avec un lot minimum autorisé.

Description

La classe CMoneyNone implémente le trading avec un lot minimum autorisé.

Déclaration

```
class CMoneyNone: public CExpertMoney
```

Titre

```
#include <Expert\Money\MoneyNone.mqh>
```

Méthodes de Classe

Initialisation	
virtual ValidationSettings	Vérifie les paramètres
Méthodes de Money et de Risk Management (Gestion du Capital et du Risque)	
virtual CheckOpenLong	Retourne le volume de trading pour une position longue
virtual CheckOpenShort	Retourne le volume de trading pour une position courte

© 2000-2015, MetaQuotes Software Corp.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La fonction retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

CheckOpenLong

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenLong(
   double  price,     // prix
   double  sl         // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne toujours la taille de lot minimale.

© 2000-2015, MetaQuotes Software Corp.

CheckOpenShort

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenShort(
   double  price,     // prix
   double  sl         // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position courte.

Note

La fonction retourne toujours la taille de lot minimale.

© 2000-2015, MetaQuotes Software Corp.

CMoneySizeOptimized

CMoneySizeOptimized est une classe implémentant un algorithme de money management, basé sur le trading avec taille variable de lot, suivant les résultats des deals précédents.

Description

CMoneySizeOptimized implémenté un algorithme de money management, basé sur le trading avec une taille variable de lot, suivant les résultats des deals précédents.

Déclaration

```
class CMoneySizeOptimized: public CExpertMoney
```

Titre

```
#include <Expert\Money\MoneySizeOptimized.mqh>
```

Méthodes de Classe

Initialisation	
DecreaseFactor	Définit la valeur du facteur de réduction
virtual ValidationSettings	Vérifie les paramètres
Méthodes de Money et de Risk Management (Gestion du Capital et du Risque)	
virtual CheckOpenLong	Retourne le volume de trading pour une position longue
virtual CheckOpenShort	Retourne le volume de trading pour une position courte

© 2000-2015, MetaQuotes Software Corp.

DecreaseFactor

Définit la valeur du facteur de réduction.

```
void  DecreaseFactor(
   double  decrease_factor      // Facteur de réduction
   )
```

Paramètres

decrease_factor

[in] Facteur de réduction.

Note

Le DecreaseFactor définit le coefficient de réduction du volume (comparé au volume de la position précédente) dans le cas de positions perdantes consécutives.

© 2000-2015, MetaQuotes Software Corp.

ValidationSettings

Vérifie les paramètres.

```
virtual bool  ValidationSettings()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si la valeur du facteur de réduction est négative, retourne faux, sinon retourne vrai.

© 2000-2015, MetaQuotes Software Corp.

CheckOpenLong

Retourne le volume de trading pour une position longue.

```
virtual double  CheckOpenLong(
   double  price,     // prix
   double  sl         // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne le volume de trading pour une position longue, le volume dépendant des résultats des deals précédents.

© 2000-2015, MetaQuotes Software Corp.

CheckOpenShort

Retourne le volume de trading pour une position courte.

```
virtual double  CheckOpenShort(
   double  price,     // prix
   double  sl         // prix du Stop Loss
   )
```

Paramètres

price

[in] Prix.

sl

[in] Prix du Stop Loss

Valeur de retour

Volume de trading pour une position longue.

Note

La fonction retourne le volume de trading pour une position courte, le volume dépendant des résultats des deals précédents.

© 2000-2015, MetaQuotes Software Corp.

Classes de Création des Panneaux de Contrôle et des Dialogues

Cette section contient les détails techniques d'utilisation des classes de création des panneaux de contrôle et une description des composants importants de la Bibliothèque Standard MQL5 (MQL5 Standard Library).

L'utilisation de ces classes vous fera gagner du temps lors de la création des panneaux de contrôle dans vos programmes MQL5 (Expert Advisors et indicateurs).

La Bibliothèque Standard MQL5 (en terme de contrôles) est située dans le répertoire de travail du terminal dans le répertoire Include\Controls.

Des exemples d'Expert Advisor illustrant l'utilisation des ces classes peuvent être trouvés dans le répertoire MQL5\Expert\Examples\Controls.

Structure auxiliaires	Description
CRect	Structure représentant une zone rectangulaire
CDateTime	Structure permettant de travailler avec la date et l'heure

Classes de base	Description
CWnd	Classe de base pour tous les contrôles
CWndObj	Classe de base pour les contrôles et les dialogues
CWndContainer	Classe de base pour les contrôles complexes (contenant des contrôles inter-dépendants)

Contrôles Simples	Description
CLabel	Contrôle, basé sur l'objet graphique "Etiquette de Texte"
CBmpButton	Contrôle, basé sur l'objet graphique "Etiquette Bitmap"
CButton	Contrôle, basé sur l'objet graphique "Bouton"
CEdit	Contrôle, basé sur l'objet graphique "Champ d'édition"
CPanel	Contrôle, basé sur l'objet graphique "Etiquette rectangulaire"
CPicture	Contrôle, basé sur l'objet graphique "Etiquette bitmap"

Contrôles complexes	Description

© 2000-2015, MetaQuotes Software Corp.

CScroll	Classe de base des barres de défilement
CScrollV	Barre de défilement verticale
CScrollH	Barre de défilement horizontale
CWndClient	Classe de base d'une zone client avec barres de défilement
CListView	Liste
CComboBox	Liste déroulante
CCheckBox	Case à cocher
CCheckGroup	Groupe de cases à cocher
CRadioButton	Bouton radio
CRadioGroup	Groupe de boutons radio
CSpinEdit	Compteur
CDialog	Dialogue
CAppDialog	Dialogue d'application

© 2000-2015, MetaQuotes Software Corp.

CRect

La classe CRect représente la zone rectangulaire du graphique.

Description

La classe CRect est la classe de la zone graphique, elle est définie par les coordonnées des coins supérieur gauche et inférieur droit d'un rectangle en coordonnées Cartésiennes.

Déclaration

```
class CRect
```

Titre

```
#include <Controls\Rect.mqh>
```

Méthodes de Classe

Propriétés	
Left	Retourne/Définit la coordonnée X du coin supérieur gauche
Top	Retourne/Définit la coordonnée Y du coin supérieur gauche
Right	Retourne/Définit la coordonnée X du coin inférieur droit
Bottom	Retourne/Définit la coordonnée Y du coin inférieur droit
Width	Retourne/Définit la largeur
Height	Retourne/Définit la hauteur
SetBound	Définit les nouvelles coordonnées en utilisant la classe CRect
Move	Définit les nouvelles coordonnées de la classe CRect
Shift	Effectue le décalage relatif des coordonnées de la classe CRect
Contains	Vérifie si le point est situé à l'intérieur de la zone de la classe CRect
Méthodes supplémentaires	
Format	Retourne les coordonnées de la zone sous forme d'une chaîne de caractères

© 2000-2015, MetaQuotes Software Corp.

Left (Méthode "Get")

Retourne la coordonnée X du coin supérieur gauche.

```
int  Left()
```

Valeur de retour

Coordonnée X du coin supérieur gauche.

Left (Méthode "Set")

Définit la coordonnée X du coin supérieur gauche.

```
void  Left(
   const int  x      // nouvelle coordonnée X
   )
```

Paramètres

x

[in] Nouvelle coordonnée X du coin supérieur gauche.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Top (Méthode "Get")

Retourne la coordonnée Y du coin supérieur gauche.

```
int  Top()
```

Valeur de retour

Coordonnée Y du coin supérieur gauche.

Top (Méthode "Set")

Définit la nouvelle coordonnée Y du coin supérieur gauche.

```
void  Top(
   const int  y      // Coordonnée Y
   )
```

Paramètres

y

[in] Nouvelle coordonnée Y du coin supérieur gauche.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Right (Méthode "Get")

Retourne la coordonnée X du coin inférieur droit.

```
int  Right()
```

Valeur de retour

Coordonnée X du coin inférieur droit.

Right (Méthode "Set")

Définit la coordonnée Y du coin inférieur droit.

```
void  Right(
   const int  x      // coordonnée x
   )
```

Paramètres

x

[in] Nouvelle coordonnée X du coin inférieur droit.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Bottom (Méthode "Get")

Retourne la coordonnée Y du coin inférieur droit.

```
int  Bottom()
```

Valeur de retour

Coordonnée Y du coin inférieur droit.

Bottom (Méthode "Set")

Définit la coordonnée Y du coin inférieur droit.

```
void  Bottom(
   const int  y      // Coordonnée Y
   )
```

Paramètres

y

[in] Nouvelle coordonnée Y du coin inférieur droit.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Width (Méthode "Get")

Retourne la largeur de la zone.

```
int  Width()
```

Valeur de retour

Largeur de la zone.

Width (Méthode "Set")

Définit la nouvelle largeur de la zone.

```
virtual bool  Width(
   const int  w      // largeur
   )
```

Paramètres

w

[in] Nouvelle largeur.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Height (Méthode "Get")

Retourne la hauteur de la zone.

```
int  Height()
```

Valeur de retour

Hauteur de la zone.

Height (Méthode "Set")

Définit la nouvelle hauteur de la zone.

```
virtual bool  Height(
   const int  h      // hauteur
   )
```

Paramètres

h

[in] Nouvelle hauteur.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

SetBound

Définit les nouvelles coordonnées de la zone utilisant les coordonnées de la classe CRect.

```
void  SetBound(
   const & CRect  rect      // Classe CRect
   )
```

Valeur de retour

Aucune.

SetBound

Définit les nouvelles coordonnées de la zone.

```
void  SetBound(
   const int  l      // ghauche
   const int  t      // dessus
   const int  r      // droit
   const int  b      // dessous
   )
```

Paramètres

l

[in] Coordonnée X du coin supérieur gauche.

t

[in] Coordonnée Y du coin supérieur gauche.

r

[in] Coordonnée X du coin inférieur droit.

b

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Move

Définit les nouvelles coordonnées de la classe CRect.

```
void  Move(
   const int  x,      // coordonnée X
   const int  y       // coordonnée Y
   )
```

Paramètres

x

[in] Nouvelle coordonnée X.

y

[in] Nouvelle coordonnée Y.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Shift

Effectue le décalage relatif des coordonnées de la classe CRect.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Paramètres

dx

[in] Delta X.

dy

[in] Delta Y.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Contains

Vérifie si le point est situé à l'intérieur de la zone de la classe CRect.

```
bool  Contains(
   const int  x,      // coordonnée X
   const int  y       // coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X.

y

[in] Coordonnée Y.

Valeur de retour

vrai si le point est situé à l'intérieur de la zone (incluant les bordures), sinon faux.

© 2000-2015, MetaQuotes Software Corp.

Format

Retourne les coordonnées de la zone sous forme d'une chaîne de caractères.

```
string  Format(
   string &  fmt,     // format
   ) const
```

Paramètres

fmt

[in] Chaîne de caractères avec le format.

Valeur de retour

Chaîne de caractères avec les coordonnées de la zone.

© 2000-2015, MetaQuotes Software Corp.

CDateTime

La classe CDateTime permet de travailler avec les dates et heures.

Description

CDateTime est une structure dérivant de MqlDateTime, elle est utilisée pour toutes les opérations utilisant des dates et des heures dans les contrôles.

Déclaration

```
struct CDateTime
```

Titre

```
#include <Tools\DateTime.mqh>
```

Méthodes de Classe

Propriétés	
MonthName	Retourne le nom du mois
ShortMonthName	Retourne le nom court du mois
DayName	Retourne le nom du jour de la semaine
ShortDayName	Retourne le nom court du jour de la semaine
DaysInMonth	Retourne le nombre de jours dans le mois
Méthodes Get/Set	
DateTime	Retourne/Définit la date/heure
Date	Définit la date
Time	Définit l'heure
Sec	Définit les secondes
Min	Définit les minutes
Hour	Définit les heures
Day	Définit le jour du mois
Mon	Définit le mois
Year	Définit l'année
Méthodes supplémentaires	
SecDec	Soustrait le nombre de secondes
SecInc	Ajoute le nombre de secondes
MinDec	Soustrait le nombre spécifié de minutes

© 2000-2015, MetaQuotes Software Corp.

MinInc	Ajoute le nombre spécifié de minutes
HourDec	Soustrait le nombre spécifié d'heures
HourInc	Ajoute le nombre spécifié d'heures
DayDec	Soustrait le nombre spécifié de jours
DayInc	Ajoute le nombre spécifié de jours
MonDec	Soustrait le nombre spécifié de mois
MonInc	Ajoute le nombre spécifié de mois
YearDec	Soustrait le nombre spécifié d'années
YearInc	Ajoute le nombre spécifié d'années

© 2000-2015, MetaQuotes Software Corp.

MonthName

Retourne le nom du mois.

```
string  MonthName()  const
```

Retourne le nom du mois par son index.

```
string  MonthName(
   const int       num          // index du mois
   )  const
```

Paramètres

num

[in] Index du mois (1-12).

Valeur de retour

Nom du mois.

© 2000-2015, MetaQuotes Software Corp.

ShortMonthName

Retourne le nom court du mois.

```
string  ShortMonthName()  const
```

Retourne le nom court du mois par son index.

```
string  ShortMonthName(
   const int       num          // index du mois
   )  const
```

Paramètres

num

[in] Index du mois (1-12).

Valeur de retour

Nom court du mois.

© 2000-2015, MetaQuotes Software Corp.

DayName

Retourne le nom du jour de la semaine.

```
string  DayName()  const
```

Retourne le nom du jour de la semaine par son index.

```
string  DayName(
   const int      num         // index du jour
   )  const
```

Paramètres

num

[in] Index du jour (de 0 à 6).

Valeur de retour

Nom du jour.

© 2000-2015, MetaQuotes Software Corp.

ShortDayName

Retourne le nom court du jour de la semaine.

```
string  ShortDayName()  const
```

Retourne le nom du jour de la semaine par son index..

```
string  ShortDayName(
   const int      num         // index du jour
   )  const
```

Paramètres

num

[in] Index du jour (de 0 à 6).

Valeur de retour

Nom court du jour.

© 2000-2015, MetaQuotes Software Corp.

DaysInMonth

Retourne le nombre de jours dans le mois

```
int  DaysInMonth()  const
```

Valeur de retour

Nombre de jours dans le mois

© 2000-2015, MetaQuotes Software Corp.

DateTime (Méthode Get)

Retourne la date et l'heure.

```
datetime  DateTime()
```

Valeur de retour

Valeur du type datetime.

DateTime (Méthode Set avec datetime)

Définit la date et l'heure à partir du type datetime.

```
void  DateTime(
   const datetime      value         // date et heure
   )
```

Paramètres

value

[in] Valeur du type datetime.

Valeur de retour

Aucune.

DateTime (Méthode Set avec MqlDateTime)

Définit la date et l'heure à partir du type MqlDateTime.

```
void  DateTime(
   const MqlDateTime   &value        // date et heure
   )
```

Paramètres

value

[in] Valeur du type MqlDateTime.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Date (Méthode Set avec datetime)

Définit la date à partir du type datetime.

```
void  Date(
   const datetime      value         // date
   )
```

Paramètres

value

[in] Valeur du type datetime.

Valeur de retour

Aucune.

Date (Méthode Set avec MqlDateTime)

Définit la date à partir du type MqlDateTime.

```
void  Date(
   const MqlDateTime   &value        // date
   )
```

Paramètres

value

[in] Valeur du type MqlDateTime.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Time (Méthode Set avec datetime)

Définit l'heure à partir du type datetime.

```
void  Time(
   const datetime      value         // heure
   )
```

Paramètres

value

[in] Valeur du type datetime.

Valeur de retour

Aucune.

Time (Méthode Set avec MqlDateTime)

Définit l'heure à partir du type MqlDateTime.

```
void  Time(
   const MqlDateTime   &value        // heure
   )
```

Paramètres

value

[in] Valeur du type MqlDateTime.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Sec

Définit les secondes.

```
void  Sec(
   const int   value        // secondes
   )
```

Paramètres

value

[in] Nombre de secondes.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Min

Définit le nombre de minutes.

```
void  Min(
   const int   value        // minutes
   )
```

Paramètres

value

[in] Nombre de minutes.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Hour

Définit l'heure.

```
void  Hour(
   const int    value        // heure
   )
```

Paramètres

value

[in] Heure.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Day

Définit le jour du mois

```
void  Day(
   const int    value        // jour
   )
```

Paramètres

value

[in] Jour du mois.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Mon

Définit le mois.

```
void  Day(
   const int    value        // mois
   )
```

Paramètres

value

[in] Numéro du mois.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Year

Définit l'année.

```
void  Day(
   const int   value        // année
   )
```

Paramètres

value

[in] Année.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

SecDec

Soustrait le nombre de secondes spécifiées.

```
void  SecDec(
   int   delta=1        // secondes
   )
```

Paramètres

delta

[in] Nombre de secondes à soustraire.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

SecInc

Ajoute le nombre spécifié de secondes

```
void  SecInc(
   int   delta=1        // secondes
   )
```

Paramètres

delta

[in] Nombre de secondes à ajouter.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

MinDec

Soustrait le nombre spécifié de minutes.

```
void  MinDec(
   int   delta=1        // minutes
   )
```

Paramètres

delta

[in] Nombre de minutes à soustraire.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

MinInc

Ajoute le nombre spécifié de minutes.

```
void  MinInc(
   int   delta=1        // minutes
   )
```

Paramètres

delta

[in] Nombre de minutes à ajouter.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

HourDec

Soustrait le nombre spécifié d'heures.

```
void  HourDec(
   int   delta=1        // heures
   )
```

Paramètres

delta

[in] Nombre d'heures à soustraire.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

HourInc

Ajoute le nombre spécifié d'heures.

```
void  HourInc(
   int   delta=1        // heures
   )
```

Paramètres

delta

[in] Nombre d'heures à ajouter.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

DayDec

Soustrait le nombre spécifié de jours.

```
void  DayDec(
   int   delta=1        // jours
   )
```

Paramètres

delta

[in] Nombre de jours à soustraire.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

DayInc

Ajoute le nombre spécifié de jours.

```
void  DayInc(
   int   delta=1        // jours
   )
```

Paramètres

delta

[in] nombre de jours à ajouter.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

MonDec

Soustrait le nombre spécifié de mois.

```
void  MonDec(
   int   delta=1        // mois
   )
```

Paramètres

delta

[in] Nombre de mois à soustraire.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

MonInc

Ajoute le nombre spécifié de mois.

```
void  MonInc(
   int   delta=1        // mois
   )
```

Paramètres

delta

[in] Nombre de mois à ajouter.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

YearDec

Soustrait le nombre spécifié d'années.

```
void  YearDec(
   int   delta=1        // années
   )
```

Paramètres

delta

[in] Nombre d"années à soustraire.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

YearInc

Ajoute le nombre spécifié d'années.

```
void  YearInc(
   int   delta=1        // années
   )
```

Paramètres

delta

[in] Nombre d'années à ajouter.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

CWnd

La classe CWnd est la classe de base de tous les contrôles inclus dans la Bilbiothèque Standard MQL5 (MQL5 Standard Library).

Description

La classe CWnd est l'implémentation de la classe du contrôle de base.

Déclaration

```
class CWnd : public CObject
```

Titre

```
#include <Controls\Wnd.mqh>
```

Méthodes de Classe

Création et destruction	
Create	Crée le contrôle
Destroy	Détruit le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
OnMouseEvent	Gestionnaire d'évènement de l'évènement CHARTEVENT_MOUSE_MOVE
Nom	
Name	Retourne le nom du contrôle
Accès au conteneur	
ControlsTotal	Retourne le nombre total de contrôles du conteneur
Control	Retourne le contrôle par son index
ControlFind	Retourne le contrôle par son identifiant
Géometrie	
Rect	Retourne le pointeur sur l'objet CRect
Left	Retourne/Définit la coordonnée X du coin supérieur gauche
Top	Retourne/Définit la coordonnée Y du coin supérieur gauche
Right	Retourne/Définit la coordonnée X du coin inférieur droit

© 2000-2015, MetaQuotes Software Corp.

Bottom	Retourne/Définit la coordonnée Y du coin inférieur droit
Width	Retourne/Définit la largeur
Height	Retourne/Définit la hauteur
Move	Définit les nouvelles coordonnées du point.
Shift	Effectue le décalage relatif des coordonnées du contrôle
Resize	Définit les nouvelles largeur/hauteur du contrôle
Contains	Vérifie si le point est situé à l'intérieur de la zone du contrôle
Alignement	
Alignment	Définit les propriétés d'alignement du contrôle
Align	Effectue l'alignement du contrôle
Identification	
Id	Retourne/Définit l'identifiant du contrôle
Etat	
IsEnabled	Retourne/Définit une valeur indiquant si le contrôle est activé ou pas
Enable	Définit une valeur indiquant si le contrôle est activé ou pas
Disable	Désactive le contrôle
IsVisible	Vérifie la visibilité du contrôle
Visible	Définit la visibilité du contrôle
Show	Affiche le contrôle
Hide	Cache le contrôle
IsActive	Vérifie si le contrôl est actif ou pas
Activate	Active le contrôle
Deactivate	Désactive le contrôle
Flags d'états	
StateFlags	Retourne/Définit les flags d'état du contrôle
StateFlagsSet	Définit les flags d'état du contrôle
StateFlagsReset	Réinitialise les flags d'état du contrôle
Flags des propriétés	
PropFlags	Retourne/Définit les flags des propriétés du

© 2000-2015, MetaQuotes Software Corp.

	contrôle
PropFlagsSet	Définit les flags des propriétés du contrôle
PropFlagsReset	Réinitialise les flags des propriétés du contrôle
HandlerOpérations de la souris	
MouseX	Retourne/Définit la coordonnée X de la souris
MouseY	Retourne/Définit la coordonnée Y de la souris
MouseFlags	Retourne/Définit l'état des boutons de la souris
MouseFocusKill	Supprime le focus de la souris
Gestionnaires d'évènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnDestroy	Gestionnaire d'évènement "Destroy"
OnMove	Gestionnaire d'évènement "Move"
OnResize	Gestionnaire d'évènement "Resize"
OnEnable	Gestionnaire d'évènement "Enable"
OnDisable	Gestionnaire d'évènement "Disable"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnActivate	Gestionnaire d'évènement "Activate"
OnDeactivate	Gestionnaire d'évènement "Deactivate"
OnClick	Gestionnaire d'évènement "Click"
OnChange	Gestionnaire d'évènement "Change"
Gestionnaires d'évènement de la souris	
OnMouseDown	Gestionnaire d'évènement "MouseDown"
OnMouseUp	Gestionnaire d'évènement "MouseUp"
Gestionnaires d'évènement de Drag n' Drop	
OnDragStart	Gestionnaire d'évènement "DragStart"
OnDragProcess	Gestionnaire d'évènement "DragProcess"
OnDragEnd	Gestionnaire d'évènement "DragEnd"
Objet de déplacement	
DragObjectCreate	Crée un objet de déplacement
DragObjectDestroy	Détruit un objet de déplacement

© 2000-2015, MetaQuotes Software Corp.

Create

Crée le contrôle.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

La méthode de la classe de base ne fait que sauvegarder les paramètres et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

Destroy

Détruit un contrôle.

```
virtual bool  Destroy()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre de l'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre de l'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnMouseEvent

Gestionnaire d'évènement de la souris (l'évènement CHARTEVENT_MOUSE_MOVE).

```
virtual bool  OnMouseEvent(
   const int  x,          // coordonnée X
   const int  y,          // coordonnée Y
   const int  flags       // flags
   )
```

Paramètres

x

[in] Coordonnée X du curseur de la souris relativement au coin supérieur gauche du graphique.

y

[in] Coordonnée Y du curseur de la souris relativement au coin supérieur gauche du graphique.

flags

[in] Flag des états des boutons de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Name

Retourne le nom du contrôle

```
string  Name()  const
```

Valeur de retour

Nom du contrôle.

© 2000-2015, MetaQuotes Software Corp.

ControlsTotal

Retourne le nombre total de contrôles du conteneur.

```
int  ControlsTotal()  const
```

Valeur de retour

Nombre de contrôles du conteneur.

Note

La méthode de la classe de base ne possède pas le conteneur, elle fournit un accès au conteneur à ses descendants et retourne toujours 0.

© 2000-2015, MetaQuotes Software Corp.

Control

Retourne le contrôle par son index

```
CWnd*  Control(
   const int  ind      // index
   )  const
```

Paramètres

ind

[in] Index du contrôle.

Valeur de retour

Un pointeur sur le contrôle.

Note

La méthode de la classe de base ne possède pas le conteneur, elle fournit un accès au conteneur à ses descendants et retourne toujours NULL.

© 2000-2015, MetaQuotes Software Corp.

ControlFind

Retourne le contrôle par son identifiant spécifié

```
virtual CWnd*  ControlFind(
   const long  id      // Identifiant
   )
```

Paramètres

id

[in] Identifiant du contrôle à trouver.

Valeur de retour

Pointeur sur le contrôle.

Note

La méthode de la classe de base ne possède pas le conteneur, elle fournit un accès au conteneur à ses descendants. Si l'identifiant spécifié correspond à l'identifiant du conteneur, elle retourne un pointeur sur elle-même (this).

© 2000-2015, MetaQuotes Software Corp.

Rect

Retourne le pointeur sur l'objet CRect.

```
const CRect*  Rect()  const
```

Valeur de retour

Retourne le pointeur sur l'objet CRect.

© 2000-2015, MetaQuotes Software Corp.

Left (Méthode "Get")

Retourne la coordonnée X du coin supérieur gauche du contrôle.

```
int  Left()
```

Valeur de retour

Coordonnée X du coin supérieur gauche du contrôle.

Left (Méthode "Set")

Définit la coordonnée X du coin supérieur gauche du contrôle.

```
void  Left(
   const int  x      // nouvelle coordonnée X
   )
```

Paramètres

x

[in] Nouvelle coordonnée X du coin supérieur gauche.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Top (Méthode "Get")

Retourne la coordonnée Y du coin supérieur gauche du contrôle.

```
int  Top()
```

Valeur de retour

Coordonnée Y du coin supérieur gauche du contrôle.

Top (Méthode "Set")

Définit la coordonnée Y du coin supérieur gauche du contrôle.

```
void  Top(
   const int  y      // Coordonnée Y
   )
```

Paramètres

y

[in] Nouvelle coordonnée Y du coin supérieur gauche.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Right (Méthode "Get")

Retourne la coordonnée Y du coin inférieur droit du contrôle.

```
int  Right()
```

Valeur de retour

Coordonnée X du coin inférieur droit.

Right (Méthode "Set")

Définit la coordonnée Y du coin inférieur droit du contrôle.

```
void  Right(
   const int  x      // coordonnée x
   )
```

Paramètres

x

[in] Nouvelle coordonnée X du coin inférieur droit.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Bottom (Méthode "Get")

Retourne la coordonnée Y du coin inférieur droit du contrôle.

```
int  Bottom()
```

Valeur de retour

La coordonnée Y du coin inférieur droit du contrôle.

Bottom (Méthode "Set")

Définit la coordonnée Y du coin inférieur droit du contrôle.

```
void  Bottom(
   const int  y      // Coordonnée Y
   )
```

Paramètres

y

[in] Nouvelle coordonnée Y du coin inférieur droit.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Width (Méthode "Get")

Retourne la largeur du contrôle.

```
int  Width()
```

Valeur de retour

Largeur du contrôle.

Width (Méthode "Set")

Définit la nouvelle largeur du contrôle.

```
virtual bool  Width(
   const int  w      // largeur
   )
```

Paramètres

w

[in] Nouvelle largeur.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Height (Méthode "Get")

Retourne la hauteur du contrôle.

```
int  Height()
```

Valeur de retour

Hauteur du contrôle.

Height (Méthode "Set")

Définit la nouvelle hauteur du contrôle.

```
virtual bool  Height(
   const int  h      // hauteur
   )
```

Paramètres

h

[in] Nouvelle hauteur.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Move

Définit les nouvelles coordonnées du contrôle.

```
void  Move(
   const int  x,      // coordonnée X
   const int  y       // coordonnée Y
   )
```

Paramètres

x

[in] Nouvelle coordonnée X.

y

[in] Nouvelle coordonnée Y.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Shift

Effectue le décalage relatif des coordonnées du contrôle.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Paramètres

dx

[in] Delta X.

dy

[in] Delta Y.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Resize

Définit les nouvelles largeur/hauteur du contrôle.

```
virtual bool  Resize(
   const int  w,      // largeur
   const int  h       // hauteur
   )
```

Paramètres

w

[in] Nouvelle largeur.

h

[in] Nouvelle hauteur.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Contains

Vérifie si le point est situé à l'intérieur de la zone du contrôle.

```
bool  Contains(
   const int  x,      // coordonnée X
   const int  y       // coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X.

y

[in] Coordonnée Y.

Valeur de retour

vrai si le point est situé à l'intérieur de la zone (incluant les bordures), sinon faux.

Contains

Vérifie si le point est situé à l'intérieur de la zone du contrôle.

```
bool  Contains(
   const CWnd*  control      // pointeur
   )  const
```

Paramètres

control

[in] Pointeur sur l'objet.

Valeur de retour

vrai si le point est situé à l'intérieur de la zone (incluant les bordures), sinon faux.

© 2000-2015, MetaQuotes Software Corp.

Alignment

Définit les paramètres d'alignement du contrôle.

```
void  Alignment(
   const int  flags,      // flags d'alignement
   const int  left,       // décalage depuis la bordure gauche
   const int  top,        // décalage depuis la bordure supérieure
   const int  right,      // décalage depuis la bordure droite
   const int  bottom      // décalage depuis la bordure inférieure
   )
```

Paramètres

flags

[in] Flags d'alignement.

left

[in] Décalage fixé depuis la bordure gauche.

top

[in] Décalage fixé depuis la bordure supérieure.

right

[in] Décalage fixé depuis la bordure droite.

bottom

[in] Décalage fixé depuis la bordure inférieure.

Valeur de retour

Aucune.

Note

Flags d'alignement :

```
enum WND_ALIGN_FLAGS
{
   WND_ALIGN_NONE=0,                                  // aucun alignement
   WND_ALIGN_LEFT=1,                                  // alignement à gauche
   WND_ALIGN_TOP=2,                                   // alignement en haut
   WND_ALIGN_RIGHT=4,                                 // alignement à droite
   WND_ALIGN_BOTTOM=8,                                // alignement en bas
   WND_ALIGN_WIDTH = WND_ALIGN_LEFT|WND_ALIGN_RIGHT,  // aligner la largeur
   WND_ALIGN_HEIGHT=WND_ALIGN_TOP|WND_ALIGN_BOTTOM,   // aligner la hauteur
   WND_ALIGN_CLIENT=WND_ALIGN_WIDTH|WND_ALIGN_HEIGHT, // aligner la hauteur et la larg
   }
```

© 2000-2015, MetaQuotes Software Corp.

Align

Effectue l'alignement du contrôle dans la zone du graphique spécifiée.

```
virtual bool  Align(
   const CRect*  rect      // pointeur
   )
```

Paramètres

rect

[in] Pointeur vers l'objet avec les coordonnées de la zone du graphique.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Les paramètres d'alignement doivent être spécifiés (aucun alignement par défaut).

© 2000-2015, MetaQuotes Software Corp.

Id (Méthode "Get")

Retourne l'identifiant du contrôle.

```
long  Id()  const
```

Valeur de retour

L'identifiant du contrôle.

Id (Méthode "Set")

Définit la nouvelle valeur de l'identifant du contrôle.

```
virtual long  Id(
   const long  id      // identifiant
   )
```

Paramètres

id

[in] Nouvelle valeur de l'identifiant du contrôle.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

IsEnabled

Retourne une valeur indiquant si le contrôle est activé ou pas

```
bool  IsEnabled()  const
```

Valeur de retour

vrai si le contrôle est actif, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

Enable

Active le contrôle.

```
virtual bool  Enable()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Si le contrôle est activé, il est capable de gérer les évènements externes.

© 2000-2015, MetaQuotes Software Corp.

Disable

Désactive le contrôle.

```
virtual bool  Disable()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Un contrôle désactivé ne permet pas de traiter les évènements externes.

© 2000-2015, MetaQuotes Software Corp.

IsVisible

Retourne une valeur indiquant si le contrôle est visible ou pas

```
bool  IsVisible()  const
```

Valeur de retour

vrai si le contrôle est affiché sur le graphique, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

Visible

Définit la visibilité du contrôle.

```
virtual bool  Visible(
   const bool  flag      // flag
   )
```

Paramètres

flag

[in] Nouveaux flag de visibilité.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Show

Affiche le contrôle.

```
virtual bool  Show()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Hide

Cache le contrôle.

```
virtual bool  Hide()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

IsActive

Retourne une valeur indiquant si le contrôle est activé ou pas.

```
bool  IsActive()  const
```

Valeur de retour

vrai si le contrôle est actif, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

Activate

Active le contrôle.

```
virtual bool  Activate()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Le contrôle devient actif lorsque le curseur de la souris passe au-dessus.

© 2000-2015, MetaQuotes Software Corp.

Deactivate

Désactive le contrôle

```
virtual bool  Deactivate()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Le contrôle devient inactif lorsque le curseur de la souris sort de la zone du contrôle.

© 2000-2015, MetaQuotes Software Corp.

StateFlags (Méthode "Get")

Retourne les flags d'état du contrôle

```
int  StateFlags()
```

Valeur de retour

Les flags d'état du contrôle.

StateFlags (Méthode "Set")

Définit les flags d'état du contrôle.

```
virtual void  StateFlags(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Nouveaux flags d'état du contrôle.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

StateFlagsSet

Définit les flags d'état du contrôle.

```
virtual void  StateFlagsSet(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Flags à définir (masque de bits).

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

StateFlagsReset

Réinitialise les flags d'état du contrôle.

```
virtual void  StateFlagsReset(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Flags à réinitialiser (masque de bits).

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

PropFlags (Méthode "Get")

Retourne les flags des propriétés du contrôle

```
void  PropFlags(
   const int  flags      // flags
   )
```

Valeur de retour

Les flags des propriétés du contrôle

PropFlags (Méthode "Set")

Définit les flags des propriétés du contrôle.

```
virtual void  PropFlags(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Nouveaux flags.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

PropFlagsSet

Définit les flags des propriétés du contrôle.

```
virtual void  PropFlagsSet(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Flags à définir (masque de bits).

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

PropFlagsReset

Réinitialise les flags des propriétés du contrôle.

```
virtual void  PropFlagsReset(
   const int  flags      // flags
   )
```

Paramètres

flags

[in] Flags à réinitialiser (masque de bits).

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

MouseX (Méthode "Set")

Sauvegarde la coordonnée X de la souris.

```
void  MouseX(
   const int  value      // coordonnée
   )
```

Paramètres

value

[in] La coordonnée X de la souris.

Valeur de retour

Aucune.

MouseX (Méthode "Get")

Retourne la coordonnée X sauvegardée de la souris.

```
int  MouseX()
```

Valeur de retour

La coordonnée X sauvegardée de la souris.

© 2000-2015, MetaQuotes Software Corp.

MouseY (Méthode "Set")

Sauvegarde la coordonnée Y de la souris.

```
void  MouseY(
   const int  value      // coordonnée
   )
```

Paramètres

value

[in] La coordonnée Y de la souris.

Valeur de retour

Aucune.

MouseY (Méthode "Get")

Retourne la coordonnée Y sauvegardée de la souris.

```
int  MouseY()
```

Valeur de retour

La coordonnée Y sauvegardée de la souris.

© 2000-2015, MetaQuotes Software Corp.

MouseFlags (Méthode "Set")

Sauvegarde l'état des boutons de la souris.

```
virtual void  MouseFlags(
   const int  value      // état
   )
```

Paramètres

value

[in] Etat des boutons de la souris.

Valeur de retour

Aucune.

MouseFlags (Méthode "Get")

Retourne l'état sauvegardé des boutons de la souris.

```
int  MouseFlags()
```

Valeur de retour

Etat des boutons de la souris.

© 2000-2015, MetaQuotes Software Corp.

MouseFocusKill

Réinitialise l'état sauvegardé des boutons de la souris et désactive le contrôle.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // identifiant
   )
```

Paramètres

id=CONTROLS_INVALID_ID

[in] Identifiant du contrôle ayant reçu le focus de la souris.

Valeur de retour

Le résultat de la désactivation du contrôle.

© 2000-2015, MetaQuotes Software Corp.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnDestroy

Le gestionnaire d'évènement "Destroy" du contrôle.

```
virtual bool  OnDestroy()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnEnable

Le gestionnaire d'évènement "Enable" (si activé, il peut répondre à l'interaction utilisateur) du contrôle.

```
virtual bool  OnEnable()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnDisable

Le gestionnaire d'évènement "Disable" (si désactivé, ne peut pas répondre à l'interaction utilisateur) du contrôle.

```
virtual bool  OnDisable()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnActivate

Le gestionnaire d'évènement "Activate" du contrôle.

```
virtual bool  OnActivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnDeactivate

Le gestionnaire d'évènement "Deactivate" du contrôle.

```
virtual bool  OnDeactivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnClick

Le gestionnaire d'évènement "Click" (clic avec le bouton gauche de la souris) du contrôle.

```
virtual bool  OnClick()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnMouseDown

Le gestionnaire d'évènement "MouseDown" du contrôle.

```
virtual bool  OnMouseDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseDown" est généré lorsque le bouton gauche de la souris est appuyé sur le contrôle.

© 2000-2015, MetaQuotes Software Corp.

OnMouseUp

Le gestionnaire d'évènement "MouseUp" du contrôle (le bouton gauche de la souris est relâché).

```
virtual bool  OnMouseUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseUp" est généré lorsque le bouton gauche de la souris est relâché au-dessus du contrôle.

© 2000-2015, MetaQuotes Software Corp.

OnDragStart

Le gestionnaire d'évènement "DragStart" du contrôle.

```
virtual bool  OnDragStart()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DragStart" est généré au démarrage du déplacement.

© 2000-2015, MetaQuotes Software Corp.

OnDragProcess

Le gestionnaire d'évènement "DragProcess" du contrôle.

```
virtual bool  OnDragProcess(
   const int  x,      // Coordonnée X
   const int  y       // Coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X actuelle du curseur de la souris.

y

[in] Coordonnée Y actuelle du curseur de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DragProcess" est généré lorsque le contrôle est déplacé.

© 2000-2015, MetaQuotes Software Corp.

OnDragEnd

Le gestionnaire d'évènement "DragEnd" du contrôle.

```
virtual bool  OnDragEnd()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DragEnd" est généré à la fin du déplacement du contrôle.

© 2000-2015, MetaQuotes Software Corp.

DragObjectCreate

Crée un objet de déplacement

```
virtual bool  DragObjectCreate()
```

Valeur de retour

vrai - en cas de succès, faux sinon

Note

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

DragObjectDestroy

Détruit un objet de déplacement

```
virtual bool  DragObjectDestroy()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CWndObj

CWndObj est une classe de base pour les contrôles simples (basés sur les objets graphiques) de la Bilibothèque Standard.

Description

La classe CWndObj implémente les méthodes de base des contrôles simples.

Déclaration

```
class CWndObj : public CWnd
```

Titre

```
#include <Controls\WndObj.mqh>
```

Méthodes de Classe

Traitement des évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
Texte	Retourne/Définit la propriété OBJPROP_TEXT de l'objet graphique
Color	Retourne/Définit la propriété OBJPROP_COLOR de l'objet graphique
ColorBackground	>Retourne/Définit la propriété OBJPROP_BGCOLOR de l'objet graphique
ColorBorder	>Retourne/Définit la propriété OBJPROP_BORDER_COLOR de l'objet graphique
Font	>Retourne/Définit la propriété OBJPROP_FONT de l'objet graphique
FontSize	>Retourne/Définit la propriété OBJPROP_FONTSIZE de l'objet graphique
ZOrder	>Retourne/Définit la propriété OBJPROP_ZORDER de l'objet graphique
Gestionnaires d'évènements des objets graphiques	
OnObjectCreate	Gestionnaire d'évènement CHARTEVENT_OBJECT_CREATE
OnObjectChange	Gestionnaire d'évènement CHARTEVENT_OBJECT_CHANGE

© 2000-2015, MetaQuotes Software Corp.

OnObjectDelete	Gestionnaire d'évènement CHARTEVENT_OBJECT_DELETE
OnObjectDrag	Gestionnaire d'évènement CHARTEVENT_OBJECT_DRAGOpérations à la souris
Gestionnaires d'évènements de changement des propriétés	
OnSetText	Gestionnaire d'évènement "SetText"
OnSetColor	Gestionnaire d'évènement "SetColor"
OnSetColorBackground	Gestionnaire d'évènement "SetColorBackground"
OnSetFont	Gestionnaire d'évènement "SetFont"
OnSetFontSize	Gestionnaire d'évènement "SetFontSize"
OnSetZOrder	Gestionnaire d'évènement "SetZOrder"
Gestionnaires d'évènements internes	
OnDestroy	Gestionnaire d'évènement "Destroy"
OnChange	Gestionnaire d'évènement "Change"

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Text (Méthode "Get")

Retourne la propriété OBJPROP_TEXT (texte) de l'objet graphique.

```
string  Text()
```

Valeur de retour

La valeur de la propriété OBJPROP_TEXT.

Text (Méthode "Set")

Définit la propriété OBJPROP_TEXT (texte) de l'objet graphique.

```
bool  Text(
   const string  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_TEXT.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Color (Méthode "Get")

Retourne la propriété OBJPROP_COLOR (couleur) de l'objet graphique.

```
color  Color()
```

Valeur de retour

La valeur de la propriété OBJPROP_COLOR.

Color (Méthode "Set")

Définit la propriété OBJPROP_COLOR (couleur) de l'objet graphique.

```
bool  Color(
   const color  value      // valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_COLOR.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ColorBackground (Méthode "Get")

Retourne la valeur de la propriété OBJPROP_BGCOLOR (couleur d'arrière plan) de l'objet graphique.

```
color  ColorBackground()
```

Valeur de retour

La valeur de la propriété OBJPROP_BGCOLOR.

ColorBackground (Méthode "Set")

Définit la propriété OBJPROP_BGCOLOR (couleur d'arrière plan) de l'objet graphique.

```
bool  ColorBackground(
   const color  value      // valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_BGCOLOR.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ColorBorder (Méthode "Get")

Retourne la valeur de la propriété OBJPROP_BORDER_COLOR (couleur de la bordure) de l'objet graphique.

```
color  ColorBorder()
```

Valeur de retour

La valeur de la propriété OBJPROP_BORDER_COLOR.

ColorBorder (Méthode "Set")

Définit la valeur de la propriété OBJPROP_BORDER_COLOR (couleur de la bordure) de l'objet graphique.

```
bool  ColorBorder(
   const color  value      // valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_BORDER_COLOR.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Font (Méthode "Get")

Retourne la propriété OBJPROP_FONT (police de caractères) de l'objet graphique.

```
string  Font()
```

Valeur de retour

La valeur de la propriété OBJPROP_FONT.

Font (Méthode "Set")

Définit la propriété OBJPROP_FONT (police de caractères) de l'objet graphique.

```
bool  Font(
   const string  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_FONT.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

FontSize (Méthode "Get")

Retourne la propriété OBJPROP_FONTSIZE (taille de la police de caractères) de l'objet graphique.

```
int  FontSize()
```

Valeur de retour

La valeur de la propriété OBJPROP_FONTSIZE.

FontSize (Méthode "Set")

Définit la propriété OBJPROP_FONTSIZE (taille de la police de caractères) de l'objet graphique.

```
bool  FontSize(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_FONTSIZE.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ZOrder (Méthode "Get")

>Retourne/Définit la propriété OBJPROP_ZORDER de l'objet graphique

```
long  ZOrder()
```

Valeur de retour

La valeur de la propriété OBJPROP_ZORDER.

ZOrder (Méthode "Set")

>Définit la propriété OBJPROP_ZORDER de l'objet graphique

```
bool  ZOrder(
   const long  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété OBJPROP_ZORDER.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnObjectCreate

Le gestionnaire d'évènement CHARTEVENT_OBJECT_CREATE de l'objet graphique.

```
virtual bool  OnObjectCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnObjectChange

Le gestionnaire d'évènement CHARTEVENT_OBJECT_CHANGE de l'objet graphique.

```
virtual bool  OnObjectChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnObjectDelete

Le gestionnaire d'évènement CHARTEVENT_OBJECT_DELETE de l'objet graphique.

```
virtual bool  OnObjectDelete()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnObjectDrag

Le gestionnaire d'évènement CHARTEVENT_OBJECT_DRAG de l'objet graphique.

```
virtual bool  OnObjectDrag()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetText

Le gestionnaire d'évènement "SetText" (changement de la propriété OBJPROP_TEXT) du contrôle.

```
virtual bool  OnSetText()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnSetColor

Le gestionnaire d'évènement "SetColor" (changement de la propriété OBJPROP_COLOR) du contrôle.

```
virtual bool  OnSetColor()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnSetColorBackground

Le gestionnaire d'évènement "SetColorBackground" (changement de la propriété OBJPROP_BGCOLOR) du contrôle.

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnSetFont

Le gestionnaire d'évènement "SetFont" (changement de la propriété OBJPROP_FONT) du contrôle.

```
virtual bool  OnSetFont()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnSetFontSize

Le gestionnaire d'évènement "SetFontSize" (changement de la propriété OBJPROP_FONTSIZE) du contrôle.

```
virtual bool  OnSetFontSize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnSetZOrder

Le gestionnaire d'évènement "SetZOrder" (changement de la propriété OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnDestroy

Le gestionnaire d'évènement "Destroy" du contrôle.

```
virtual bool  OnDestroy()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CWndContainer

La classe CWndContainer est une classe de base pour les contrôles complexes (contenant des contrôles inter-dépendants) de la Bilbiothèque Standard MQL5 (MQL5 Standard Library).

Description

La classe CWndContainer implémente les méthodes de base des contrôles complexes.

Déclaration

```
class CWndContainer : public CWnd
```

Titre

```
#include <Controls\WndContainer.mqh>
```

Méthodes de Classe

Destruction	
Destroy	Détruit tous les contrôles du conteneur
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
OnMouseEvent	Le gestionnaire d'évènement CHARTEVENT_MOUSE_MOVE
Accès au conteneur	
ControlsTotal	Retourne le nombre total de contrôles dans le conteneur
Control	Retourne un contrôle par son index
ControlFind	Retourne un contrôle par son identifiant
Ajout/Suppression	
Add	Ajoute un contrôle dans le conteneur
Delete	Supprime un contrôle du conteneur
Géometrie	
Move	Définit les nouvelles coordonnées de tous les contrôles du conteneur.
Shift	Effectue le décalage relatif des coordonnées de tous les contrôles du conteneur.
Identification	
Id	Définit l'identifiant de tous les contrôles du conteneur

© 2000-2015, MetaQuotes Software Corp.

Etat	
Enable	Active tous les contrôles du conteneur
Disable	Désactive tous les contrôles du conteneur
Show	Affiche tous les contrôles du conteneur
Hide	Cache tous les contrôles du conteneur
HandlerOpérations de la souris	
MouseFocusKill	Supprime le focus de la souris
Opérations avec des fichiers	
Save	Sauvegarde les informations du conteneur dans un fichier
Load	Charge les informations du conteneur depuis un fichier
Gestionnaires d'évènements internes	
OnResize	Gestionnaire d'évènement "Resize"
OnActivate	Gestionnaire d'évènement "Activate"
OnDeactivate	Gestionnaire d'évènement "Deactivate"

© 2000-2015, MetaQuotes Software Corp.

Destroy

Détruit tous les contrôles du conteneur.

```
virtual bool  Destroy()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnMouseEvent

Gestionnaire d'évènements de la souris

```
virtual bool  OnMouseEvent(
   const int  x,          // coordonnée X
   const int  y,          // coordonnée Y
   const int  flags       // flags
   )
```

Paramètres

x

[in] Coordonnée X du curseur de la souris relativement au coin supérieur gauche du graphique.

y

[in] Coordonnée Y du curseur de la souris relativement au coin supérieur gauche du graphique.

flags

[in] Flag des états des boutons de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ControlsTotal

Retourne le nombre total de contrôles du conteneur.

```
int  ControlsTotal()  const
```

Valeur de retour

Nombre de contrôles dans le conteneur.

© 2000-2015, MetaQuotes Software Corp.

Control

Retourne un contrôle par son index.

```
CWnd*  Control(
   const int  ind      // index
   )  const
```

Paramètres

ind

[in] Index du contrôle.

Valeur de retour

Pointeur sur le contrôle, NULL si le contrôle n'est pas trouvé.

© 2000-2015, MetaQuotes Software Corp.

ControlFind

Retourne un contrôle du conteneur par son identifiant.

```
virtual CWnd*  ControlFind(
   const long  id      // identifiant
   )
```

Paramètres

id

[in] Identificant du contrôle.

Valeur de retour

Pointeur sur le contrôle, NULL si le contrôle n'est pas trouvé.

© 2000-2015, MetaQuotes Software Corp.

Add

Ajoute un contrôle dans le conteneur.

```
bool  Add(
   CWnd&  control      // référence
   )
```

Paramètres

control

[in] Contrôle à ajouter, passé par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Delete

Supprime un contrôle du conteneur.

```
bool  Delete(
   CWnd&  control      // référence
   )
```

Paramètres

control

[in] Contrôle à supprimer, passé par référence.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Move

Définit les nouvelles coordonnées de tous les contrôles du conteneur.

```
virtual bool  Move(
   const int  x,      // Coordonnée X
   const int  y       // Coordonnée Y
   )
```

Paramètres

x

[in] Nouvelle coordonnée X du coin supérieur gauche.

y

[in] Nouvelle coordonnée Y du coin supérieur gauche.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Shift

Effectue le décalage relatif des coordonnées de tous les contrôles du conteneur.

```
virtual bool  Shift(
   const int  dx,      // décalage x
   const int  dy       // décalage y
   )
```

Paramètres

dx

[in] Delta X.

dy

[in] Delta Y.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Id

Définit l'identifiant de tous les contrôles du conteneur.

```
virtual long  Id(
   const long  id      // identifiant
   )
```

Paramètres

id

[in] Identificant du groupe de base.

Valeur de retour

Nombre d'identifiants, utilisé par les contrôles du conteneur.

© 2000-2015, MetaQuotes Software Corp.

Enable

Active tous les contrôles du conteneur.

```
virtual bool  Enable()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Disable

Désactive tous les contrôles du conteneur.

```
virtual bool  Disable()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Show

Affiche tous les contrôles du conteneur.

```
virtual bool  Show()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Hide

Cache tous les contrôles du conteneur.

```
virtual bool  Hide()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

MouseFocusKill

Réinitialise l'état sauvegardé des boutons de la souris et désactive tous les contrôles du conteneur.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // identifiant
   )
```

Paramètres

id=CONTROLS_INVALID_ID

[in] Identifiant du contrôle ayant reçu le focus de la souris.

Valeur de retour

Le résultat de la désactivation des contrôles.

© 2000-2015, MetaQuotes Software Corp.

Save

Sauvegarde les informations du conteneur dans un fichier

```
virtual bool  Save(
   const int  file_handle      // handle
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire (doit être ouvert en écriture).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Load

Charge les informations du conteneur depuis un fichier

```
virtual bool  Load(
   const int  file_handle      // handle
   )
```

Paramètres

file_handle

[in] Handle du fichier binaire (doit être ouvert en lecture).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnActivate

Le gestionnaire d'évènement "Activate" du contrôle.

```
virtual bool  OnActivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnDeactivate

Le gestionnaire d'évènement "Deactivate" du contrôle.

```
virtual bool  OnDeactivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

CLabel

La classe CLabel est une classe de contrôle simple, basé sur l'objet graphique "Etiquette de texte".

Description

La classe CLabel permet la création d'étiquettes de texte simples.

Déclaration

```
class CLabel : public CWndObj
```

Titre

```
#include <Controls\Label.mqh>
```

Méthodes de Classe

Create	
Création	Crée le contrôle
Gestionnaires d'évènements de changement des propriétés	
OnSetText	Gestionnaire d'évènement "SetText"
OnSetColor	Gestionnaire d'évènement "SetColor"
OnSetFont	Gestionnaire d'évènement "SetFont"
OnSetFontSize	Gestionnaire d'évènement "SetFontSize"
Gestionnaires d'évènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnMove	Gestionnaire d'évènement "Move"

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CLabel.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnSetText

Le gestionnaire d'évènement "SetText" (changement de la propriété OBJPROP_TEXT) du contrôle.

```
virtual bool  OnSetText()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetColor

Le gestionnaire d'évènement "SetColor" (changement de la propriété OBJPROP_COLOR) du contrôle.

```
virtual bool  OnSetColor()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetFont

Le gestionnaire d'évènement "SetFont" (changement de la propriété OBJPROP_FONT) du contrôle.

```
virtual bool  OnSetFont()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetFontSize

Le gestionnaire d'évènement "SetFontSize" (changement de la propriété OBJPROP_FONTSIZE) du contrôle.

```
virtual bool  OnSetFontSize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CBmpButton

La classe CBmpButton est une classe de contrôle simple, basé sur l'objet graphique "Etiquette Bitmap".

Description

La classe CBmpButton permet la création de boutons avec une image.

Déclaration

```
class CBmpButton : public CWndObj
```

Titre

```
#include <Controls\BmpButton.mqh>
```

Méthodes de Classe

Create	
Create	Crée le contrôle
Propriétés	
Border	Retourne/Définit la propriété "Border" du contrôle
BmpNames	Définit le nom des fichiers bmp du contrôle
BmpOffName	Retourne/Définit le nom du fichier bmp pour l'état Eteint (Off)
BmpOnName	Retourne/Définit le nom du fichier bmp pour l'état Allumé (On)
BmpPassiveName	Retourne/Définit le nom du fichier bmp pour l'état passif
BmpActiveName	Retourne/Définit le nom du fichier bmp pour l'état actif
Etat	
Pressed	Retourne/Définit l'état du contrôle
Locking	Retourne/Définit la propriété "Locking" du contrôle
Gestionnaires d'évènements internes	
OnSetZOrder	Gestionnaire d'évènement "SetZOrder"
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"

© 2000-2015, MetaQuotes Software Corp.

OnMove	Gestionnaire d'évènement "Move"
OnChange	Gestionnaire d'évènement "Change"
OnActivate	Gestionnaire d'évènement "Activate"
OnDeactivate	Gestionnaire d'évènement "Deactivate"
OnMouseDown	Gestionnaire d'évènement "MouseDown"
OnMouseUp	Gestionnaire d'évènement "MouseUp"

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CBmpButton.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Border (Méthode "Get")

Retourne la propriété "Border" (épaisseur de la bordure) du contrôle.

```
int  Border()  const
```

Valeur de retour

La propriété "Border".

Border (Méthode "Set")

Définit la propriété "Border" (épaisseur de la bordure) du contrôle.

```
bool  Border(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "Border".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

BmpNames

Définit le nom des fichiers bmp du contrôle

```
bool  BmpNames(
   const string  off="",      // nom du fichier
   const string  on=""        // nom du fichier
   )
```

Paramètres

off=""

[in] Nom du fichier bmp pour l'état Eteint (OFF).

on=""

[in] Nom du fichier bmp pour l'état Allumé (ON).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

BmpOffName (Méthode "Get")

Retourne le nom du fichier bmp pour l'"état Eteint (OFF).

```
string  BmpOffName()  const
```

Valeur de retour

Nom du fichier bmp pour l'état Eteint (OFF).

BmpOffName (Méthode "Set")

Définit le nom du fichier bmp pour l'état Eteint (OFF).

```
bool  BmpOffName(
   const string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nom du fichier bmp pour l'état Eteint (OFF).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

BmpOnName (Méthode "Get")

Retourne le nom du fichier bmp pour l'état Allumé (ON).

```
string  BmpOnName()  const
```

Valeur de retour

Nom du fichier bmp pour l'état Allumé (ON).

BmpOnName (Méthode "Set")

Définit le nom du fichier bmp pour l'état Allumé (ON).

```
bool  BmpOnName(
   const string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nom du fichier bmp pour l'état Allumé (ON).

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

BmpPassiveName (Méthode "Get")

Retourne le nom du fichier bmp pour l'état passif du contrôle.

```
string  BmpPassiveName()  const
```

Valeur de retour

Nom du fichier bmp pour l'état passif du contrôle.

BmpPassiveName (Méthode "Set")

Définit le nom du fichier bmp pour l'état passif.

```
bool  BmpPassiveName(
   const string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nom du fichier bmp pour l'état passif du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

BmpActiveName (Méthode "Get")

Retourne le nom du fichier bmp pour l'état actif.

```
string  BmpActiveName()  const
```

Valeur de retour

Nom du fichier bmp pour l'état actif.

Note

Le contrôle devient actif lorsque le curseur de la souris passe au-dessus.

BmpActiveName (Méthode "Set")

Définit le nom du fichier bmp pour l'état actif

```
bool  BmpActiveName(
   const string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nom du fichier bmp pour l'état actif.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Pressed (Méthode "Get")

Retourne l'état (propriété "Pressed") du contrôle.

```
bool  Pressed()  const
```

Valeur de retour

Etat du contrôle.

Pressed (Méthode "Set")

Définit l'état (propriété "Pressed") du contrôle.

```
bool  Pressed(
   const bool  pressed      // nouvel état
   )
```

Paramètres

pressed

[in] Nouvel état du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Locking (Méthode "Get")

Retourne la propriété "Locking" du contrôle.

```
bool  Locking()  const
```

Valeur de retour

Valeur de la propriété "Locking".

Locking (Méthode "Set")

Définit la nouvelle valeur de la propriété "Locking" du contrôle.

```
void  Locking(
   const bool  locking      // nouvelle valeur
   )
```

Paramètres

locking

[in] Nouvelle valeur pour la propriété "Locking".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

OnSetZOrder

Le gestionnaire d'évènement "SetZOrder" (changement de la propriété OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnActivate

Le gestionnaire d'évènement "Activate" du contrôle.

```
virtual bool  OnActivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnDeactivate

Le gestionnaire d'évènement "Deactivate" du contrôle.

```
virtual bool  OnDeactivate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnMouseDown

Le gestionnaire d'évènement "MouseDown" du contrôle.

```
virtual bool  OnMouseDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseDown" est généré lorsque le bouton gauche de la souris est appuyé sur le contrôle.

© 2000-2015, MetaQuotes Software Corp.

OnMouseUp

Le gestionnaire d'évènement "MouseUp" du contrôle (le bouton gauche de la souris est relâché).

```
virtual bool  OnMouseUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseUp" est généré lorsque le bouton gauche de la souris est relâché au-dessus du contrôle.

© 2000-2015, MetaQuotes Software Corp.

CButton

La classe CButton est une classe de contrôle simple, basé sur l'objet graphique "Bouton".

Description

La classe CButton est une classe pour la création de boutons simples.

Déclaration

```
class CButton : public CWndObj
```

Titre

```
#include <Controls\Button.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Etat	
Pressed	Retourne/Définit la propriété "Pressed"
Locking	Retourne/Définit la propriété "Locking"
Gestionnaires d'évènements de changement des propriétés	
OnSetText	Gestionnaire d'évènement "SetText"
OnSetColor	Gestionnaire d'évènement "SetColor"
OnSetColorBackground	Gestionnaire d'évènement "SetColorBackground"
OnSetColorBorder	Gestionnaire d'évènement "SetColorBorder"
OnSetFont	Gestionnaire d'évènement "SetFont"
OnSetFontSize	Gestionnaire d'évènement "SetFontSize"
Gestionnaires d'évènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnMove	Gestionnaire d'évènement "Move"
OnResize	Gestionnaire d'évènement "Resize"
OnMouseDown	Gestionnaire d'évènement "MouseDown"
OnOnMouseUp	Gestionnaire d'évènement "MouseUp"

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CButton.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Pressed (Méthode "Get")

Retourne l'état (propriété "Pressed") du contrôle.

```
bool  Pressed()  const
```

Valeur de retour

Etat du contrôle.

Pressed (Méthode "Set")

Définit l'état (propriété "Pressed") du contrôle.

```
bool  Pressed(
   const bool  pressed      // nouvel état
   )
```

Paramètres

pressed

[in] Nouvel état du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Locking (Méthode "Get")

Retourne la propriété "Locking" du contrôle.

```
bool  Locking()  const
```

Valeur de retour

Valeur de la propriété "Locking".

Locking (Méthode "Set")

Définit la nouvelle valeur de la propriété "Locking" du contrôle.

```
void  Locking(
   const bool  locking      // nouvelle valeur
   )
```

Paramètres

locking

[in] Nouvelle valeur pour la propriété "Locking".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

OnSetText

Le gestionnaire d'évènement "SetText" (changement de la propriété OBJPROP_TEXT) du contrôle.

```
virtual bool  OnSetText()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetColor

Le gestionnaire d'évènement "SetColor" (changement de la propriété OBJPROP_COLOR) du contrôle.

```
virtual bool  OnSetColor()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetColorBackground

Le gestionnaire d'évènement "SetColorBackground" (changement de la propriété OBJPROP_BGCOLOR) du contrôle.

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetColorBorder

Le gestionnaire d'évènement "SetColorBorder" (changement de la propriété OBJPROP_BORDER_COLOR) du contrôle.

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetFont

Le gestionnaire d'évènement "SetFont" (changement de la propriété OBJPROP_FONT) du contrôle.

```
virtual bool  OnSetFont()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetFontSize

Le gestionnaire d'évènement "SetFontSize" (changement de la propriété OBJPROP_FONTSIZE) du contrôle.

```
virtual bool  OnSetFontSize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnMouseDown

Le gestionnaire d'évènement "MouseDown" du contrôle.

```
virtual bool  OnMouseDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseDown" est généré lorsque le bouton gauche de la souris est appuyé sur le contrôle.

© 2000-2015, MetaQuotes Software Corp.

OnMouseUp

Le gestionnaire d'évènement "MouseUp" du contrôle (le bouton gauche de la souris est relâché).

```
virtual bool  OnMouseUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "MouseUp" est généré lorsque le bouton gauche de la souris est relâché au-dessus du contrôle.

© 2000-2015, MetaQuotes Software Corp.

CEdit

La classe CEdit est une classe de contrôle simple, basé sur l'objet graphique "Edit".

Description

La classe CButton est une classe pour la création de contrôles dans lesquels l'utilisateur peut saisir un texte.

Déclaration

```
class CEdit : public CWndObj
```

Titre

```
#include <Controls\Edit.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Propriétés	
ReadOnly	Retourne/Définit la propriété "ReadOnly"
TextAlign	Retourne/Definit la propriété "TextAlign"
Gestionnaires d'évènements des objets graphiques	
OnObjectEndEdit	Gestionnaire (virtuel) de l'évènement CHARTEVENT_OBJECT_ENDEDIT
Gestionnaires d'évènements de changement des propriétés	
OnSetText	Gestionnaire d'évènement "SetText"
OnSetColor	Gestionnaire d'évènement "SetColor"
OnSetColorBackground	Gestionnaire d'évènement "SetColorBackground"
OnSetColorBorder	Gestionnaire d'évènement "SetColorBorder"
OnSetFont	Gestionnaire d'évènement "SetFont"
OnSetFontSize	Gestionnaire d'évènement "SetFontSize"
OnSetZOrder	Gestionnaire d'évènement "SetZOrder"
Gestionnaires d'évènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"

© 2000-2015, MetaQuotes Software Corp.

OnHide	Gestionnaire d'évènement "Hide"
OnMove	Gestionnaire d'évènement "Move"
OnResize	Gestionnaire d'évènement "Resize"
OnChange	Gestionnaire d'évènement "Change"
OnClick	Gestionnaire d'évènement "Click"

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CEdit.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ReadOnly (Méthode "Get")

Retourne la propriété "ReadOnly" du contrôle.

```
bool  ReadOnly()
```

Valeur de retour

La valeur de la propriété "ReadOnly".

ReadOnly (Méthode "Set")

Définit la valeur de la propriété "ReadOnly" du contrôle.

```
bool  ReadOnly(
   const bool  flag      // nouvelle valeur
   )
```

Paramètres

flag

[in] Nouvelle valeur de la propriété "ReadOnly".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

TextAlign (Méthode "Get")

Retourne la valeur de la propriété "TextAlign" (mode d'alignement du texte).

```
ENUM_ALIGN_MODE  TextAlign() const
```

Valeur de retour

Définit la valeur de la propriété "TextAlign" du contrôle.

TextAlign (Méthode "Set")

Définit la nouvelle valeur de la propriété "TextAlign" (mode d'alignement du texte) du contrôle.

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // nouvelle valeur
   )
```

Paramètres

align

[in] Nouvelle valeur de la propriété "TextAlign".

Valeur de retour

vrai si réalisé avec succès, faux si la propriété n'a pas été changé.

© 2000-2015, MetaQuotes Software Corp.

OnObjectEndEdit

Gestionnaire de l'évènement CHARTEVENT_OBJECT_ENDEDIT

```
virtual bool  OnObjectEndEdit()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetText

Le gestionnaire d'évènement "SetText" (changement de la propriété OBJPROP_TEXT) du contrôle.

```
virtual bool  OnSetText()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetColor

Le gestionnaire d'évènement "SetColor" (changement de la propriété OBJPROP_COLOR) du contrôle.

```
virtual bool  OnSetColor()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetColorBackground

Le gestionnaire d'évènement "SetColorBackground" (changement de la propriété OBJPROP_BGCOLOR) du contrôle.

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetColorBorder

Le gestionnaire d'évènement "SetColorBorder" (changement de la propriété OBJPROP_BORDER_COLOR<t3>) du contrôle.</t3>

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetFont

Le gestionnaire d'évènement "SetFont" (changement de la propriété OBJPROP_FONT) du contrôle.

```
virtual bool  OnSetFont()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetFontSize

Le gestionnaire d'évènement "SetFontSize" (changement de la propriété OBJPROP_FONTSIZE) du contrôle.

```
virtual bool  OnSetFontSize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetZOrder

Le gestionnaire d'évènement "SetZOrder" (changement de la propriété OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnClick

Le gestionnaire d'évènement "Click" (clic avec le bouton gauche de la souris) du contrôle.

```
virtual bool  OnClick()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CPanel

La classe CPanel est une classe de contrôle simple, basé sur l'objet graphique "Etiquette rectangulaire".

Description

La classe CPanel permet de combiner des contrôles ayant des fonctions similaires dans le groupe.

Déclaration

```
class CPanel : public CWndObj
```

Titre

```
#include <Controls\Panel.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Propriétés de l'objet graphique	
BorderType	Retourne la propriété "BorderType" de l'objet graphique
Gestionnaires d'évènements des objets graphiques	
OnSetText	Gestionnaire d'évènement "SetText"
OnSetColorBackground	Gestionnaire d'évènement "SetColorBackground"
OnSetColorBorder	Gestionnaire d'évènement "SetColorBorder"
Gestionnaires d'évènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnMove	Gestionnaire d'évènement "Move"
OnResize	Gestionnaire d'évènement "Resize"
OnChange	Gestionnaire d'évènement "Change"

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CPanel.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

BorderType (Méthode "Get")

Retourne la propriété "BorderType" de l'objet graphique.

```
ENUM_BORDER_TYPE  BorderType()
```

Valeur de retour

La valeur de la propriété "BorderType".

BorderType (Méthode "Set")

Définit la nouvelle valeur de la propriété "BorderType" de l'objet graphique.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // valeur
   )
```

Paramètres

type

[in] Nouvelle valeur de la propriété "BorderType".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnSetText

Le gestionnaire d'évènement "SetText" (changement de la propriété OBJPROP_TEXT) du contrôle.

```
virtual bool  OnSetText()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetColorBackground

Le gestionnaire d'évènement "SetColorBackground" (changement de la propriété OBJPROP_BGCOLOR) du contrôle.

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnSetColorBorder

Le gestionnaire d'évènement "SetColorBorder" (changement de la propriété OBJPROP_BORDER_COLOR<t3>) du contrôle.</t3>

```
virtual bool  OnSetColorBackground()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CPicture

La classe CPicture est une classe de contrôle simple, basé sur l'objet graphique "Etiquette Bitmap".

Description

La classe CPicture permet la création d'images graphiques simples.

Déclaration

```
class CPicture : public CWndObj
```

Titre

```
#include <Controls\Picture.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Propriétés de l'objet graphique	
Border	Retourne/Définit l'épaisseur du contour de l'objet graphique
BmpName	Retourne/Définit le nom du fichier bmp du contrôle
Evènements internes	
OnCreate	Gestionnaire d'évènement "Create"
OnShow	Gestionnaire d'évènement "Show"
OnHide	Gestionnaire d'évènement "Hide"
OnMove	Gestionnaire d'évènement "Move"
OnChange	Gestionnaire d'évènement "Change"

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CPicture.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Border (Méthode "Get")

Retourne la propriété "Border" (épaisseur de la bordure) du contrôle.

```
int  Border()  const
```

Valeur de retour

La propriété "Border".

Border (Méthode "Set")

Définit la propriété "Border" (épaisseur de la bordure) du contrôle.

```
bool  Border(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "Border".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

BmpName (Méthode "Get")

Retourne le nom du fichier bmp du contrôle.

```
string  BmpName()  const
```

Valeur de retour

Nom du fichier bmp du contrôle.

BmpName (Méthode "Set")

Définit le nom du fichier bmp du contrôle.

```
bool  BmpName(
   const string  name      // nom du fichier
   )
```

Paramètres

name

[in] Nom du fichier bmp du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnCreate

Le gestionnaire d'évènement "Create" du contrôle.

```
virtual bool  OnCreate()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnMove

Le gestionnaire d'évènement "Move" du contrôle.

```
virtual bool  OnMove()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChange

Le gestionnaire d'évènement "Change" du contrôle.

```
virtual bool  OnChange()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CScroll

La classe CCanvas est la classe de base pour créer des barres de défilement.

Description

CScroll est un contrôle complexe (contenant différents contrôles), contenant la fonctionnalité de base de création des barres de défilement. La classe de base elle-même n'est pas utilisée comme un contrôle séparé, mais deux de ses descendants (classes CScrollV et CScrollH) le sont.

Déclaration

```
class CScroll : public CWndContainer
```

Titre

```
#include <Controls\Scrolls.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements des objets graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
MinPos	Retourne/Définit la position minimale
MaxPos	Retourne/Définit la position maximale
CurrPos	Retourne/Définit la position courante
Création des contrôles dépendants	
CreateBack	Crée le bouton d'arrière plan
CreateInc	Crée le bouton d'incrémentation de la barre de défilement
CreateDec	Crée le bouton de décrémentation de la barre de défilement
CreateThumb	Crée l'ascenseur (pouvant être déplacé) de la barre de défilement
Gestionnaires d'évènements des contrôles dépendants	
OnClickInc	Gestionnaire d'évènement utilisé pour gérer les évènements du bouton d'incrémentation

© 2000-2015, MetaQuotes Software Corp.

OnClickDec	Gestionnaire d'évènement utilisé pour gérer les évènements du boutonu de décrémentation
Gestionnaires d'évènements internes	
OnShow	Gestionnaire d'évènement "Create"
OnHide	Gestionnaire d'évènement "Hide"
OnChangePos	Gestionnaire d'évènement "ChangePosition"
Gestionnaires de déplacement de l'ascenseur	
OnThumbDragStart	Gestionnaire d'évènement "ThumbDragStart"
OnThumbDragProcess	Gestionnaire d'évènement "ThumbDragProcess"
OnThumbDragEnd	Gestionnaire d'évènement "ThumbDragEnd"
Position	
CalcPos	Retourne la position de la barre de défilement par ses coordonnées

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CScroll.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

MinPos (Méthode "Get")

Retourne la valeur de la propriété "MinPos" (position minimale) du contrôle CScroll.

```
int  MinPos()  const
```

Valeur de retour

Nouvelle valeur de la propriété "MinPos".

MinPos (Méthode "Set")

Définit la valeur de la propriété "MinPos" (position minimale) du contrôle CScroll.

```
void  MinPos(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "MinPos".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

MaxPos (Méthode "Get")

Définit la valeur de la propriété "MaxPos" (position maximale) du contrôle CScroll.

```
int  MaxPos()  const
```

Valeur de retour

Nouvelle valeur de la propriété "MaxPos".

MaxPos (Méthode "Set")

Définit la valeur de la propriété "MaxPos" (position maximale) du contrôle CScroll.

```
void  MaxPos(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriéé "MaxPos".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

CurrPos (Méthode "Get")

Retourne la valeur de la propriété "CurrPos" (position actuelle) du contrôle CScroll.

```
int  CurrPos()  const
```

Valeur de retour

[in] Nouvelle valeur de la propriété "CurrPos".

CurrPos (Méthode "Set")

Définit la valeur de la propriété "CurrPos" (position actuelle) du contrôle CScroll.

```
void  CurrPos(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "Caption".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

CreateBack

Crée le bouton d'arrière plan du contrôle CScroll.

```
virtual bool  CreateBack()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateInc

Crée le bouton d'incrémentation du contrôle CScroll.

```
virtual bool  CreateInc()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateDec

Crée le bouton de décrémentation du contrôle CScroll.

```
virtual bool  CreateDec()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateThumb

Crée l'ascenseur (pouvant être déplacé) du contrôle CScroll

```
virtual bool  CreateThumb()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnClickInc

Le gestionnaire d'évènement "ClickInc" (clic avec le bouton gauche de la souris sur le bouton d'incrémentation) du contrôle.

```
virtual bool  OnClickInc()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnClickDec

Le gestionnaire d'évènement "ClickDec" (clic avec le bouton gauche de la souris sur le bouton de décrémentation) du contrôle.

```
virtual bool  OnClickDec()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnShow

Le gestionnaire d'évènement "Show" du contrôle.

```
virtual bool  OnShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnHide

Le gestionnaire d'évènement "Hide" du contrôle.

```
virtual bool  OnHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChangePos

Le gestionnaire d'évènement "ChangePos" (changement de position) du contrôle.

```
virtual bool  OnChangePos()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnThumbDragStart

Le gestionnaire d'évènement "ThumbDragStart" (début du déplacement de l'ascenceur) du contrôle.

```
virtual bool  OnThumbDragStart()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragStart" est généré au démarrage du déplacement de l'ascenseur.

© 2000-2015, MetaQuotes Software Corp.

OnThumbDragProcess

Le gestionnaire d'évènement "ThumbDragProcess" du contrôle.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // Coordonnée X
   const int  y       // Coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X actuelle du curseur de la souris.

y

[in] Coordonnée Y actuelle du curseur de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragProcess" est généré lorsque le contrôle de la barre de défilement (l'ascenseur) est déplacé.

© 2000-2015, MetaQuotes Software Corp.

OnThumbDragEnd

Le gestionnaire d'évènement "ThumbDragEnd" (fin du déplacement de l'ascenseur) du contrôle.

```
virtual bool  OnThumbDragEnd()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragEnd" est généré lorsque le déplacement du contrôle de la barre de défilement (l'ascenseur) est terminé.

© 2000-2015, MetaQuotes Software Corp.

CalcPos

Retourne la position de la barre de défilement par ses coordonnées

```
virtual int  CalcPos(
   const int      coord      // coordonnée
   )
```

Paramètres

coord

[in] Coordonnée de la barre de défilement.

Valeur de retour

Position de la barre de défilement.

© 2000-2015, MetaQuotes Software Corp.

CScrollV

La classe CScrollV est la classe du contrôle complexe "Barre de défilement verticale".

Description

La classe CScrollV permet la création des barres de défilement verticale.

Déclaration

```
class CScrollV : public CScroll
```

Titre

```
#include <Controls\Scrolls.mqh>
```

Méthodes de Classe

Contrôles dépendants	
CreateInc	Cré le bouton d'incrémentation de la barre de défilement
CreateDec	Cré le bouton de décrémentation de la barre de défilement
CreateThumb	Crée le bouton de l'ascenseur (pouvant être déplacé)
Gestionnaires d'évènements internes	
OnResize	Gestionnaire d'évènement "Resize"
OnChangePos	Gestionnaire d'évènement "ChangePosition"
Gestionnaires d'évènement de Drag n' Drop	
OnThumbDragStart	Gestionnaire d'évènement "ThumbDragStart"
OnThumbDragProcess	Gestionnaire d'évènement "ThumbDragProcess"
OnThumbDragEnd	Gestionnaire d'évènement "ThumbDragEnd"
Position	
CalcPos	Retourne la position de la barre de défilement par ses coordonnées

© 2000-2015, MetaQuotes Software Corp.

CreateInc

Crée le bouton d'incrémentation du contrôle.

```
virtual bool  CreateInc()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateDec

Crée le bouton de décrémentation du contrôle.

```
virtual bool  CreateDec()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateThumb

Crée l'ascenseur (pouvant être déplacé) du contrôle.

```
virtual bool  CreateThumb()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChangePos

Le gestionnaire d'évènement "ChangePos" (changement de position) du contrôle.

```
virtual bool  OnChangePos()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnThumbDragStart

Le gestionnaire d'évènement "ThumbDragStart" (début du déplacement de l'ascenceur) du contrôle.

```
virtual bool  OnThumbDragStart()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragStart" est généré au démarrage du déplacement de l'ascenseur.

© 2000-2015, MetaQuotes Software Corp.

OnThumbDragProcess

Le gestionnaire d'évènement "ThumbDragProcess" du contrôle.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // Coordonnée X
   const int  y       // Coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X actuelle du curseur de la souris.

y

[in] Coordonnée Y actuelle du curseur de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragProcess" est généré lorsque le contrôle de la barre de défilement (l'ascenseur) est déplacé.

© 2000-2015, MetaQuotes Software Corp.

OnThumbDragEnd

Le gestionnaire d'évènement "ThumbDragEnd" (fin du déplacement de l'ascenseur) du contrôle.

```
virtual bool  OnThumbDragEnd()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragEnd" est généré lorsque le déplacement du contrôle de la barre de défilement (l'ascenseur) est terminé.

© 2000-2015, MetaQuotes Software Corp.

CalcPos

Retourne la position de la barre de défilement par ses coordonnées

```
virtual int  CalcPos(
   const int      coord      // coordonnée
   )
```

Paramètres

coord

[in] Coordonnée de la barre de défilement.

Valeur de retour

Position de la barre de défilement.

© 2000-2015, MetaQuotes Software Corp.

CScrollH

La classe CScrollH est la classe du contrôle complexe "Barre de défilement horizontale".

Description

La classe CScrollH permet la création des barres de défilement horizontale.

Déclaration

```
class CScrollH : public CScroll
```

Titre

```
#include <Controls\Scrolls.mqh>
```

Méthodes de Classe

Contrôles dépendants	
CreateInc	Cré le bouton d'incrémentation de la barre de défilement
CreateDec	Cré le bouton de décrémentation de la barre de défilement
CreateThumb	Crée le bouton de l'ascenseur (pouvant être déplacé)
Gestionnaires d'évènements internes	
OnResize	Gestionnaire d'évènement "Resize"
OnChangePos	Gestionnaire d'évènement "ChangePosition"
Gestionnaires d'évènement de Drag n' Drop	
OnThumbDragStart	Gestionnaire d'évènement "ThumbDragStart"
OnThumbDragProcess	Gestionnaire d'évènement "ThumbDragProcess"
OnThumbDragEnd	Gestionnaire d'évènement "ThumbDragEnd"
Position	
CalcPos	Retourne la position de la barre de défilement par ses coordonnées

© 2000-2015, MetaQuotes Software Corp.

CreateInc

Crée le bouton d'incrémentation du contrôle.

```
virtual bool  CreateInc()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateDec

Crée le bouton de décrémentation du contrôle.

```
virtual bool  CreateDec()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateThumb

Crée l'ascenseur (pouvant être déplacé) du contrôle.

```
virtual bool  CreateThumb()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChangePos

Le gestionnaire d'évènement "ChangePos" (changement de position) du contrôle.

```
virtual bool  OnChangePos()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnThumbDragStart

Le gestionnaire d'évènement "ThumbDragStart" (début du déplacement de l'ascenceur) du contrôle.

```
virtual bool  OnThumbDragStart()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragStart" est généré au démarrage du déplacement de l'ascenseur.

© 2000-2015, MetaQuotes Software Corp.

OnThumbDragProcess

Le gestionnaire d'évènement "ThumbDragProcess" du contrôle.

```
virtual bool  OnThumbDragProcess(
   const int  x,     // Coordonnée X
   const int  y      // Coordonnée Y
   )
```

Paramètres

x

[in] Coordonnée X actuelle du curseur de la souris.

y

[in] Coordonnée Y actuelle du curseur de la souris.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragProcess" est généré lorsque le contrôle de la barre de défilement (l'ascenseur) est déplacé.

© 2000-2015, MetaQuotes Software Corp.

OnThumbDragEnd

Le gestionnaire d'évènement "ThumbDragEnd" (fin du déplacement de l'ascenseur) du contrôle.

```
virtual bool  OnThumbDragEnd()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "ThumbDragEnd" est généré lorsque le déplacement du contrôle de la barre de défilement (l'ascenseur) est terminé.

© 2000-2015, MetaQuotes Software Corp.

CalcPos

Retourne la position de la barre de défilement par ses coordonnées

```
virtual int  CalcPos(
   const int      coord      // coordonnée
   )
```

Paramètres

coord

[in] Coordonnée de la barre de défilement.

Valeur de retour

Position de la barre de défilement.

© 2000-2015, MetaQuotes Software Corp.

CWndClient

CWndClient est une classe du contrôle complexe de "Zone client" (avec des contrôles inter-dépendants). C'est la classe de base pour la création des zones de défilement des barres de défilement.

Description

CWndClient implémente la fonctionnalité de création des zone client avec des barres de défilement.

Déclaration

```
class CWndClient : public CWndContainer
```

Titre

```
#include <Controls\WndClient.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaire d'évènement graphique.	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
ColorBackground	Définit la couleur d'arrière plan.
ColorBorder	Définit la couleur de la bordure.
BorderType	Définit le type de bordure.
Paramètres	
VScrolled	Retourne/Définit le flag indiquant que la barre de défilement verticale est utilisée
HScrolled	Retourne/Définit le flag indiquant que la barre de défilement horizontale est utilisée
Contrôles dépendants	
CreateBack	Cée l'arrière plan de la barre de défilement
CreateScrollV	Crée la barre de défilement verticale
CreateScrollH	Crée la barre de défilement horizontale
Gestionnaires d'évènements internes	
OnResize	Gestionnaire d'évènement "Resize"
Gestionnaires d'évènements des contrôles	

© 2000-2015, MetaQuotes Software Corp.

dépendants	
OnVScrollShow	Gestionnaire (virtuel) de l'évènement "Show" du contrôle dépendant VScroll
OnVScrollHide	Gestionnaire (virtuel) de l'évènement "Hide" du contrôle dépendant VScroll
OnHScrollShow	Gestionnaire (virtuel) de l'évènement "Show" du contrôle dépendant HScroll
OnHScrollHide	Gestionnaire (virtuel) de l'évènement "Hide" du contrôle dépendant HScroll
OnScrollLineDown	Gestionnaire (virtuel) de l'évènement "ScrollLineDown" du contrôle dépendant VScroll
OnScrollLineUp	Gestionnaire (virtuel) de l'évènement "ScrollLineUp" du contrôle dépendant VScroll
OnScrollLineLeft	Gestionnaire (virtuel) de l'évènement "ScrollLineLeft" du contrôle dépendant HScroll
OnScrollLineRight	Gestionnaire (virtuel) de l'évènement "ScrollLineRight" du contrôle dépendant HScroll
Resize	
Rebound	Définit les nouvelles coordonnées du contrôle en utilisant les coordonnées de la classe CRect

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CWndClient.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ColorBackground

Définit la couleur d'arrière plan du contrôle.

```
bool  ColorBackground(
   const color  value      // nouvelle couleur
   )
```

Paramètres

value

[in] Nouvelle couleur d'arrière plan du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ColorBorder

Définit la couleur de bordure du contrôle.

```
bool  ColorBorder(
   const color  value      // couleur
   )
```

Paramètres

value

[in] Nouvelle couleur de bordure du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

BorderType

Définit le type de bordure du contrôle.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // type de bordure
   )
```

Paramètres

type

[in] Type de bordure du contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

VScrolled (Méthode "Get")

Retourne/Définit le flag indiquant que la barre de défilement verticale est utilisée.

```
bool  VScrolled()
```

Valeur de retour

vrai si la barre de défilement verticale est utilisée, faux sinon.

VScrolled (Méthode "Set")

Définit le flag indiquant que la barre de défilement verticale est utilisée.

```
bool  VScrolled(
   const bool  flag      // flag
   )
```

Paramètres

flag

[in] Flag.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

HScrolled (Méthode "Get")

Retourne/Définit le flag indiquant que la barre de défilement horizontale est utilisée.

```
bool  HScrolled()
```

Valeur de retour

vrai si la barre de défilement horizontal est utilisée, faux sinon.

HScrolled (Méthode "Set")

Définit le flag indiquant que la barre de défilement horizontale est utilisée.

```
bool  HScrolled(
   const bool  flag      // flag
   )
```

Paramètres

flag

[in] Flag.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateBack

Crée le bouton d'arrière plan du contrôle.

```
virtual bool  CreateBack()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateScrollV

Crée la barre de défilement verticale

```
virtual bool  CreateScrollV()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateScrollH

Crée la barre de défilement horizontale.

```
virtual bool  CreateScrollH()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnVScrollShow

Le gestionnaire d'évènement "VScrollShow" (affichage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnVScrollHide

Le gestionnaire d'évènement "VScrollHide" (masquage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnHScrollShow

Le gestionnaire d'évènement "HScrollShow" (affichage de la barre de défilement horizontale) du contrôle.

```
virtual bool  OnHScrollShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnHScrollHide

Le gestionnaire d'évènement "HScrollHide" (masquage de la barre de défilement horizontale) du contrôle.

```
virtual bool  OnHScrollHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnScrollLineDown

Le gestionnaire d'évènement "ScrollLineDown" (défilement vertical vers le bas) du contrôle.

```
virtual bool  OnScrollLineDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnScrollLineUp

Le gestionnaire d'évènement "ScrollLineUp" (défilement vertical vers le haut) du contrôle.

```
virtual bool  OnScrollLineUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnScrollLineLeft

Le gestionnaire d'évènement "ScrollLineLeft" (défilement vers la gauche) du contrôle.

```
virtual bool  OnScrollLineLeft()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

OnScrollLineRight

Le gestionnaire d'évènement "ScrollLineRight" (défilement vers la droite) du contrôle.

```
virtual bool  OnScrollLineRight()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

La méthode de la classe de base n'effectue aucune action et retourne toujours vrai.

© 2000-2015, MetaQuotes Software Corp.

ReBound

Définit les nouvelles coordonnées du contrôle utilisant les coordonnées de la classe CRect.

```
void  ReBound(
   const & CRect  rect      // Classe CRect
   )
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

CListView

La classe CListView est une classe de contrôle complexe de ListView (avec des contrôles inter-dépendants).

Description

La classe CListView encapsule les fonctionnalités de contrôle de liste.

Déclaration

```
class CListView : public CWndClient
```

Titre

```
#include <Controls\ListView.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Paramètres	
TotalView	Définit le nombre d'éléments affichés dans le contrôle
Ajout/Suppression	
AddItem	Ajoute un élément
Données	
Select	Sélectionne l'élément courant par son index
SelectByText	Séléection l'élément courant par son texte
SelectByValue	Séléection l'élément courant par sa valeur
Données en lecture seule	
Value	Retourne la valeur de l'élément courant de la liste
Contrôles dépendants	
CreateRow	Crée une ligne dans la liste ListView
Gestionnaires d'évènements internes	
OnResize	Gestionnaire (virtuel) de l'évènement "Resize"

© 2000-2015, MetaQuotes Software Corp.

Gestionnaires d'évènements des contrôles dépendants	
OnVScrollShow	Gestionnaire (virtuel) de l'évènement "Show" du contrôle dépendant VScroll
OnVScrollHide	Gestionnaire (virtuel) de l'évènement "Hide" du contrôle dépendant VScroll
OnScrollLineDown	Gestionnaire (virtuel) de l'évènement "ScrollLineDown" du contrôle dépendant VScroll
OnScrollLineUp	Gestionnaire (virtuel) de l'évènement "ScrollLineUp" du contrôle dépendant VScroll
OnItemClick	Gestionnaire (virtuel) de l'évènement "ItemClick"
Rafraîchissement	
Redraw	Redessine le contrôle
RowState	Définit l'état de la ligne spécifiée
CheckView	Vérifie la visibilité de la ligne spécifiée

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CListView.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

TotalView

Définit le nombre d'éléments affichés dans le contrôle

```
bool  TotalView(
   const int  value      // éléments affichés
   )
```

Paramètres

value

[in] Le nombre d'éléments affichés dans le contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

Note

Le nombre d'éléments affichés ne peut être spécifié qu'une seule fois.

© 2000-2015, MetaQuotes Software Corp.

AddItem

Ajoute un élément au contrôle.

```
bool  AddItem(
   const string  item,     // texte
   const long    value     // valeur
   )
```

Paramètres

item

[in] Texte.

value

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Select

Sélectionne l'élément courant par son index.

```
bool  Select(
   const int  index     // index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

SelectByText

Séléection l'élément courant par son texte

```
bool  SelectByText(
   const string  text     // texte
   )
```

Paramètres

text

[in] Texte.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

SelectByValue

Séléection l'élément courant par sa valeur

```
bool  SelectByValue(
   const long  value     // valeur
   )
```

Paramètres

value

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Value

Retourne la valeur de l'élément courant de la liste.

```
long  Value()
```

Valeur de retour

La valeur de l'élément courant de la liste.

© 2000-2015, MetaQuotes Software Corp.

CreateRow

Crée une ligne dans le contrôle ListView

```
bool  CreateRow(
   const int  index     // index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnResize

Le gestionnaire d'évènement "Resize" du contrôle.

```
virtual bool  OnResize()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnVScrollShow

Le gestionnaire d'évènement "VScrollShow" (affichage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnVScrollHide

Le gestionnaire d'évènement "VScrollHide" (masquage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnScrollLineDown

Le gestionnaire d'évènement "ScrollLineDown" (défilement vertical vers le bas) du contrôle.

```
virtual bool  OnScrollLineDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnScrollLineUp

Le gestionnaire d'évènement "ScrollLineUp" (défilement vertical vers le haut) du contrôle.

```
virtual bool  OnScrollLineUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnItemClick

Le gestionnaire d'évènement "ItemClick" (clic de souris sur une ligne) du contrôle.

```
virtual bool  OnItemClick()
   const int    index     // index de la ligne
   )
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

Redraw

Redessine le contrôle.

```
bool  Redraw()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

RowState

Définit l'état de la ligne spécifiée.

```
bool  RowState(
   const int   index      // index
   const bool  select     // état
   )
```

Paramètres

index

[in] Index de la ligne.

select

[in] Etat de la ligne.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CheckView

Vérifie la visibilité de la ligne spécifiée.

```
bool  CheckView()
```

Valeur de retour

vrai - si la ligne sélectionnée est visible, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CComboBox

La classe CComboBox est une classe de contrôle complexe de ComboBox (avec des contrôles inter-dépendants).

Description

Le contrôle ComboBox est constitué d'une liste et d'un contrôle statique et est prévu pour la sélection. La partie 'liste' du contrôle se déroule lorsque l'utilisateur clique sur la flèche située à côté du contrôle

Déclaration

```
class CComboBox : public CWndContainer
```

Titre

```
#include <Controls\ComboBox.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Add	
AddItem	Ajoute un élément
Paramètres	
ListViewItems	Définit le nombre d'éléments affichés dans le contrôle
Données	
Select	Sélectionne l'élément courant par son index
SelectByText	Séléection l'élément courant par son texte
SelectByValue	Séléection l'élément courant par sa valeur
Données en lecture seule	
Value	Retourne la valeur de l'élément courant de la liste
Contrôles dépendants	
CreateEdit	Crée un contrôle dépendant (zone d'édition)
CreateButton	Crée un contrôle dépendant (bouton)
CreateList	Crée un contrôle dépendant (liste)

© 2000-2015, MetaQuotes Software Corp.

Gestionnaires d'évènements des contrôles dépendants	
OnClickEdit	Gestionnaire (virtuel) de l'évènement "ClickEdit"
OnClickButton	Gestionnaire (virtuel) de l'évènement "ClickButton"
OnChangeList	Gestionnaire (virtuel) de l'évènement "ChangeList"
Affichage/Masquage	
ListShow	Affiche la liste des éléments
ListHide	Cache la liste des éléments

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CComboBox.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

AddItem

Ajoute un élément au contrôle.

```
bool  AddItem(
   const string  item,     // texte
   const long    value     // valeur
   )
```

Paramètres

item

[in] Texte.

value=0

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ListViewItems

Définit le nombre d'éléments affichés dans le contrôle CComboBox.

```
void  ListViewItems(
   const int     value      // nombre d'éléments
   )
```

Paramètres

value

[in] Nombre d'éléments dans la liste.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Select

Sélectionne l'élément courant par son index.

```
bool  Select(
   const int  index     // index
   )
```

Paramètres

index

[in] Index de l'élément.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

SelectByText

Séléection l'élément courant par son texte

```
bool  SelectByText(
   const string  text     // texte
   )
```

Paramètres

text

[in] Texte de l'élément.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

SelectByValue

Séléection l'élément courant par sa valeur

```
bool  SelectByValue(
   const long  value     // valeur
   )
```

Paramètres

value

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Value

Retourne la valeur de l'élément courant de la liste.

```
long  Value()
```

Valeur de retour

La valeur de l'élément courant de la liste.

© 2000-2015, MetaQuotes Software Corp.

CreateEdit

Crée un contrôle dépendant (zone d'édition)

```
virtual bool  CreateEdit()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateButton

Crée un contrôle dépendant (bouton).

```
virtual bool  CreateButton()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateList

Crée un contrôle dépendant (liste)

```
virtual bool  CreateList()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnClickEdit

Le gestionnaire d'évènement "ClickEdit" (clic de souris sur la zone d'édition) du contrôle.

```
virtual bool  OnClickEdit()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnClickButton

Le gestionnaire d'évènement "ClickButton" (clic de souris sur le bouton) du contrôle.

```
virtual bool  OnClickButton()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChangeList

Le gestionnaire d'évènement "ChangeList" (changement dans la liste).

```
virtual bool  OnChangeList()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

ListShow

Affiche la liste des éléments.

```
virtual bool  ListShow()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ListHide

Cache la liste des éléments.

```
virtual bool  ListHide()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CCheckBox

La classe CCheckBox est une classe de contrôle complexe de CheckBox.

Description

Le contrôle CCheckBox affiche une case à cocher permettant à l'utilisateur de sélectionner une condition vraie ou fausse.

Déclaration

```
class CCheckBox : public CWndContainer
```

Titre

```
#include <Controls\CheckBox.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
Texte	Retourne/Définit le texte de l'étiquette associée au contrôle
Color	Retourne/Définit la couleur du texte de l'étiquette associée au contrôle
Etat	
Checked	Retourne/Définit une valeur indiquant si le contrôle est coché ou pas
Données	
Value	Retourne/Définit la valeur associée au contrôle
Contrôles dépendants	
CreateButton	Crée un contrôle dépendant (bouton)
CreateLabel	Crée un contrôle dépendant (étiquette)
Gestionnaires d'évènements des contrôles dépendants	
ClickButton	Gestionnaire (virtuel) de l'évènement "ClickButton"
ClickLabel	Gestionnaire (virtuel) de l'évènement

© 2000-2015, MetaQuotes Software Corp.

	"ClickLabel"

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CCheckBox.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Text (Méthode "Get")

Retourne le texte de l'étiquette associée au contrôle.

```
string  Text()
```

Valeur de retour

Texte de l'étiquette.

Text (Méthode "Set")

Définit le texte de l'étiquette associée au contrôle.

```
bool  Text(
   const string  value      // texte
   )
```

Paramètres

value

[in] Nouveau texte de l'étiquette.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Color (Méthode "Get")

Retourne la couleur de l'étiquette associée au contrôle.

```
color  Color()  const
```

Valeur de retour

Couleur de l'étiquette.

Color (Méthode "Set")

Définit la couleur de l'étiquette associée au contrôle.

```
bool  Color(
   const color  value      // couleur
   )
```

Paramètres

value

[in] Nouvelle couleur d'étiquette.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Checked (Méthode "Get")

Retourne l'état du contrôle

```
bool  Checked()  const
```

Valeur de retour

Etat du contrôle.

Checked (Méthode "Set")

Définit l'état du contrôle.

```
bool  Checked(
   const bool  flag      // état
   )
```

Paramètres

flag

[in] Nouvel état.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Value (Méthode "Get")

Retourne la valeur associée au contrôle.

```
int  Value()  const
```

Valeur de retour

La valeur associée au contrôle.

Value (Méthode "Set")

Définit la valeur associée au contrôle.

```
void  Value(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

CreateButton

Crée un contrôle dépendant (bouton).

```
virtual bool  CreateButton()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateLabel

Crée un contrôle dépendant (étiquette)

```
virtual bool  CreateLabel()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnClickButton

Le gestionnaire d'évènement "ClickButton" (clic de souris sur le bouton) du contrôle.

```
virtual bool  OnClickButton()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnClickLabel

Le gestionnaire d'évènement "ClickLabel" (clic de souris sur l'étiquette) du contrôle.

```
virtual bool  OnClickLabel()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CCheckGroup

La classe CCheckGroup est une classe de contrôle complexe de CheckGroup (avec des contrôles inter-dépendants).

Description

La classe CCheckGroup permet de créer des contrôles permettant d'afficher et d'éditer des flags.

Déclaration

```
class CCheckGroup : public CWndClient
```

Titre

```
#include <Controls\CheckGroup.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement de tous les évènements graphiques
Add	
AddItem	Ajoute un nouvel élément
Données en lecture seule	
Value	Retourne la valeur associée au contrôle
Contrôles dépendants	
CreateButton	Crée un nouvel élément CCheckBox
Gestionnaires d'évènements des contrôles dépendants	
OnVScrollShow	Gestionnaire (virtuel) de l'évènement "Show" du contrôle dépendant VScroll
OnVScrollHide	Gestionnaire (virtuel) de l'évènement "Hide" du contrôle dépendant VScroll
OnScrollLineDown	Gestionnaire (virtuel) de l'évènement "ScrollLineUp" du contrôle dépendant VScroll
OnScrollLineUp	Gestionnaire (virtuel) de l'évènement "ScrollLineDown" du contrôle dépendant VScroll
OnChangeItem	Gestionnaire (virtuel) de l'évènement "ChangeItem" du contrôle dépendant VScroll

© 2000-2015, MetaQuotes Software Corp.

Rafraichissement	
Redraw	Redessine le groupe
RowState	Définit l'état de l'élément spécifié

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CCheckGroup.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

AddItem

Ajoute un élément au contrôle.

```
bool  AddItem(
   const string  item,     // texte
   const long    value     // valeur
   )
```

Paramètres

item

[in] Texte.

value=0

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Value

Retourne la valeur associée au contrôle.

```
long  Value()
```

Valeur de retour

La valeur associée au contrôle.

Note

La valeur dépend des états de tous les éléments du groupe CCheckGroup.

© 2000-2015, MetaQuotes Software Corp.

CreateButton

Crée une nouvelle instance de la classe CCheckBox à l'index spécifié.

```
bool  CreateButton(
   int  index      // index
   )
```

Paramètres

index

[in] Index du nouvel élément au sein du CCheckGroup.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnVScrollShow

Le gestionnaire d'évènement "VScrollShow" (affichage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnVScrollHide

Le gestionnaire d'évènement "VScrollHide" (masquage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnScrollLineDown

Le gestionnaire d'évènement "ScrollLineDown" (défilement vertical vers le bas) du contrôle.

```
virtual bool  OnScrollLineDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnScrollLineUp

Le gestionnaire d'évènement "ScrollLineUp" (défilement vertical vers le haut) du contrôle.

```
virtual bool  OnScrollLineUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChangeItem

Le gestionnaire d'évènement "ChangeItem" du contrôle.

```
virtual bool  OnChangeItem(
   const int  index      // index
   )
```

Paramètres

index

[in] Index de l'élément modifié.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

Redraw

Redessine le contrôle.

```
bool  Redraw()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

RowState

Définit l'état de l'élément spécifié.

```
bool  RowState(
   const int   index,      // index de l'élément
   const bool  select      // état
   )
```

Paramètres

index

[in] Index de l'élément à modifier.

select

[in] Nouvel état.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CRadioButton

La classe CRadioButton est une classe de contrôle complexe de RadioButton.

Description

La classe CRadioButton elle-même n'est pas utilisée, elle est utilisée pour la création des éléments de type CRadioGroup.

Déclaration

```
class CRadioButton : public CWndContainer
```

Titre

```
#include <Controls\RadioButton.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement de tous les évènements graphiques
Propriétés	
Texte	Retourne/Définit le texte de l'étiquette associée au contrôle
Color	Retourne/Définit la couleur du texte de l'étiquette associée au contrôle
Etat	
Etat	Retourne/Définit l'état
Contrôles dépendants	
CreateButton	Crée un bouton
CreateLabel	Crée une étiquette
Gestionnaires d'évènements des contrôles dépendants	
OnClickButton	Gestionnaire (virtuel) de l'évènement "ClickButton"
OnClickLabel	Gestionnaire (virtuel) de l'évènement "ClickLabel"

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un contrôle CRadioButton.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Text (Méthode "Get")

Retourne le texte de l'étiquette associée au contrôle.

```
string  Text()
```

Valeur de retour

Texte de l'étiquette.

Text (Méthode "Set")

Définit le texte de l'étiquette associée au contrôle.

```
bool  Text(
   const string  value      // texte
   )
```

Paramètres

value

[in] Nouveau texte de l'étiquette.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Color (Méthode "Get")

Retourne la couleur de l'étiquette associée au contrôle.

```
color  Color()  const
```

Valeur de retour

Couleur de l'étiquette.

Color (Méthode "Set")

Définit la couleur de l'étiquette associée au contrôle.

```
bool  Color(
   const color  value      // couleur
   )
```

Paramètres

value

[in] Nouvelle couleur d'étiquette.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

State (Méthode "Get")

Retourne l'état du bouton.

```
bool  State()  const
```

Valeur de retour

Etat du bouton.

State (Méthode "Set")

Définit l'état du bouton.

```
bool  State(
   const bool  flag      // flag
   )
```

Paramètres

flag

[in] Nouvel état du bouton.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateButton

Crée un bouton.

```
virtual bool  CreateButton()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateLabel

Crée une étiquette.

```
virtual bool  CreateLabel()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnClickButton

Le gestionnaire d'évènement "ClickButton" (clic de souris) du contrôle.

```
virtual bool  OnClickButton()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnClickLabel

Le gestionnaire d'évènement "ClickLabel" (clic de souris sur l'étiquette) du contrôle.

```
virtual bool  OnClickLabel()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CRadioGroup

La classe CRadioGroup est une classe de contrôle complexe de RadioGroup (avec des contrôles inter-dépendants).

Description

La classe CRadioGroup permet à l'utilisateur de sélectionner une seule option d'un groupe de choix parmi différents contrôles CRadioButton.

Déclaration

```
class CRadioGroup : public CWndClient
```

Titre

```
#include <Controls\RadioGroup.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Add	
AddItem	Ajoute un nouvel élément
Données en lecture seule	
Value	Retourne la valeur associée au contrôle
Contrôles dépendants	
CreateButton	Crée un nouvel élément CRadioButton
Gestionnaires d'évènements des contrôles dépendants	
OnVScrollShow	Gestionnaire (virtuel) de l'évènement "Show" du contrôle dépendant VScroll
OnVScrollHide	Gestionnaire (virtuel) de l'évènement "Hide" du contrôle dépendant VScroll
OnScrollLineDown	Handler (virtuel) de l'évènement "ScrollLineDown" du contrôle dépendant VScroll
OnScrollLineUp	Gestionnaire (virtuel) de l'évènement "ScrollLineUp" du contrôle dépendant VScroll
OnChangeItem	Gestionnaire (virtuel) de l'évènement

© 2000-2015, MetaQuotes Software Corp.

	"ChangeItem"
Rafraîchissement	
Redraw	Redessine le groupe d'éléments
RowState	Définit l'état de l'élément spécifié
Select	Sélectionne l'élément courant

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CRadioGroup.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

AddItem

Ajoute un élément au contrôle.

```
bool  AddItem(
   const string  item,     // texte
   const long    value=0   // valeur
   )
```

Paramètres

item

[in] Texte.

value=0

[in] Valeur de type long.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Value

Retourne la valeur associée au contrôle.

```
long  Value()
```

Valeur de retour

La valeur associée au contrôle.

Note

La valeur dépend de l'état de tous les éléments CRadioButton du contrôle CRadioGroup.

© 2000-2015, MetaQuotes Software Corp.

CreateButton

Crée une nouvelle instance de classe CRadioButton à l'index spécifié.

```
bool  CreateButton(
   const int  index      // index
   )
```

Paramètres

index

[in] Index du nouvel élément dans le groupe CRadioGroup.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnVScrollShow

Le gestionnaire d'évènement "VScrollShow" (affichage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollShow()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnVScrollHide

Le gestionnaire d'évènement "VScrollHide" (masquage de la barre de défilement vertical) du contrôle.

```
virtual bool  OnVScrollHide()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnScrollLineDown

Le gestionnaire d'évènement "ScrollLineDown" (défilement vertical vers le bas) du contrôle.

```
virtual bool  OnScrollLineDown()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnScrollLineUp

Le gestionnaire d'évènement "ScrollLineUp" (défilement vertical vers le haut) du contrôle.

```
virtual bool  OnScrollLineUp()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChangeItem

Le gestionnaire d'évènement "ChangeItem" du contrôle.

```
virtual bool  OnChangeItem(
   const int  index      // index
   )
```

Paramètres

index

[in] Index de l'élément modifié.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

Redraw

Redessine le contrôle.

```
bool  Redraw()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

RowState

Définit l'état de l'élément spécifié.

```
bool  RowState(
   const int   index,      // index de l'élément
   const bool  select      // état
   )
```

Paramètres

index

[in] Index de l'élément à modifier.

select

[in] Nouvel état.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Select

Sélectionne l'élément courant.

```
void  Select(
   const int  index      // index
   )
```

Paramètres

index

[in] Index de l'élément à sélectionner.

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

CSpinEdit

La classe CSpinEdit est la classe du contrôle complexe SpinEdit (avec des contrôles inter-dépendants).

Description

La classe CSpinEdit permet la création de contrôles permettant d'éditer des valeurs de type integer. La valeur est incrémentée automatiquement lors de l'appui sur le bouton haut, et décrémentée en appuyant sur le bouton bas.

Déclaration

```
class CSpinEdit : public CWndContainer
```

Titre

```
#include <Controls\SpinEdit.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
MinValue	Retourne/Définit la position minimale permise
MaxValue	Retourne/Définit la position maximale permise
Etat	
Value	Retourne/Définit la valeur courante
Contrôles dépendants	
CreateEdit	Crée un contrôle dépendant (zone d'édition)
CreateInc	Crée un contrôle dépendant (bouton haut)
CreateDec	Crée un contrôle dépendant (bouton bas)
Gestionnaires d'évènements des contrôles dépendants	
OnClickInc	Gestionnaire (virtuel) de l'évènement "ClickInc"
OnClickDec	Gestionnaire (virtuel) de l'évènement "ClickDec"
Gestionnaires d'évènements internes	
OnChangeValue	Gestionnaire (virtuel) de l'évènement "ChangeValue"

© 2000-2015, MetaQuotes Software Corp.

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CSpinEdit.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

MinValue (Méthode "Get")

Retourne la valeur de la propriété "MinValue" (valeur minimale permise) du contrôle.

```
int  MinValue()  const
```

Valeur de retour

La valeur de la propriété "MinValue".

MinValue (Méthode "Set")

Définit la valeur de la propriété "MinValue" (valeur minmale permise) du contrôle.

```
void  MinValue(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "MinValue".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

MaxValue (Méthode "Get")

Retourne la valeur de la propriété "MaxValue" (valeur maximale permise) du contrôle.

```
int  MaxValue()  const
```

Valeur de retour

La valeur de la propriété "MaxValue".

MaxValue (Méthode "Set")

Définit la valeur de la propriété "MaxValue" (valeur maximale permise) du contrôle.

```
void  MaxValue(
   const int  value      // nouvelle valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "MaxValue".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Value (Méthode "Get")

Retourne la valeur de la propriété "Value" (valeur actuelle) du contrôle.

```
int  Value()  const
```

Valeur de retour

La valeur de la propriété "Value".

Value (Méthode "Set")

Définit la valeur de la propriété "Value" (valeur actuelle) du contrôle.

```
void  Value(
   const int  value      // valeur
   )
```

Paramètres

value

[in] Nouvelle valeur de la propriété "Value".

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

CreateEdit

Crée un contrôle dépendant (CEdit).

```
virtual bool  CreateEdit()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateInc

Crée un contrôle dépendant (bouton haut)

```
virtual bool  CreateInc()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateDec

Crée un contrôle dépendant (bouton bas).

```
virtual bool  CreateDec()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnClickInc

Le gestionnaire d'évènement "ClickInc" (clic de la souris sur le bouton d'incrémentation) du contrôle.

```
virtual bool  OnClickInc()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnClickDec

Le gestionnaire d'évènement "ClickDec" (clic de la souris sur le bouton de décrémentation) du contrôle.

```
virtual bool  OnClickDec()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

OnChangeValue

Le gestionnaire d'évènement "ChangeValue" du contrôle.

```
virtual bool  OnChangeValue()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

© 2000-2015, MetaQuotes Software Corp.

CDialog

La classe CDialog est une classe de contrôle complexe de Dialog.

Description

La classe CAppDialog permet de combiner des contrôles ayant des fonctions différentes dans un groupe au sein d'un groupe.

Déclaration

```
class CDialog : public CWndContainer
```

Titre

```
#include <Controls\Dialog.mqh>
```

Méthodes de Classe

Création	
Create	Crée le contrôle
Gestionnaires d'évènements graphiques	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Propriétés	
Caption	Retourne/Définit la valeur de la propriété "Caption"
Add	
Add	Ajoute un contrôle dans la zone client
Contrôles dépendants	
CreateWhiteBorder	Crée un contrôle dépendant (bordure blanche)
CreateBackground	Crée un contrôle dépendant (arrière plan)
CreateCaption	Crée un contrôle dépendant (légende)
CreateButtonClose	Crée un contrôle dépendant (bouton Fermer)
CreateClientArea	Crée un contrôle dépendant (zone client)
Gestionnaires d'évènements des contrôles dépendants	
OnClickCaption	Gestionnaire d'évènement "ClickCaption"
OnClickButtonClose	Gestionnaire d'évènement "ClickButtonClose"
Accés à la zone client	

© 2000-2015, MetaQuotes Software Corp.

ClientAreaVisible	Définit une valeur indiquant si la zone client est visible ou pas
ClientAreaLeft	Retourne la coordonnée X du coin supérieur gauche de la zone client du contrôle
ClientAreaTop	Retourne la coordonnée Y du coin supérieur gauche de la zone client du contrôle
ClientAreaRight	Retourne la coordonnée X du coin inférieur droit de la zone client du contrôle
ClientAreaBottom	Retourne la coordonnée Y du coin inférieur droit de la zone client du contrôle
ClientAreaWidth	Retourne la largeur de la zone client
ClientAreaHeight	Retourne la hauteur de la zone client
Gestionnaires d'évènement de Drag n' Drop	
OnDialogDragStart	Gestionnaire (virtuel) de l'évènement "DialogDragStart"
OnDialogDragProcess	Gestionnaire (virtuel) de l'évènement "DialogDragProcess"
OnDialogDragEnd	Gestionnaire (virtuel) de l'évènement "DialogDragEnd"

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CDialog.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Caption (Méthode "Get")

Retourne la propriété "Caption" du contrôle CDialog.

```
string  MinValue()  const
```

Valeur de retour

La propriété "Caption".

Caption (Méthode "Set")

Définit la propriété "Caption" du contrôle CDialog.

```
bool  Caption(
   const string  text      // texte
   )
```

Paramètres

text

[in] Nouvelle valeur de la propriété "Caption".

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Add

Ajoute le contrôle dans la zone client par pointeur.

```
bool  Add(
   CWnd   *control,         // pointeur
   )
```

Paramètres

control

[in] Pointeur vers le contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

Add

Ajoute le contrôle dans la zone client par référence.

```
bool  Add(
   CWnd   &control,         // référence
   )
```

Paramètres

control

[in] Référence sur le contrôle.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateWhiteBorder

Crée un contrôle dépendant (bordure blanche).

```
virtual bool  CreateWhiteBorder()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateBackground

Crée un contrôle dépendant (arrière plan).

```
virtual bool  CreateBackground()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateCaption

Crée un contrôle dépendant (légende).

```
virtual bool  CreateCaption()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateButtonClose

Crée un contrôle dépendant (bouton Fermer)

```
virtual bool  CreateButtonClose()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateClientArea

Crée un contrôle dépendant (zone client)

```
virtual bool  CreateClientArea()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnClickCaption

Le gestionnaire d'évènement "ClickCaption" du contrôle.

```
virtual bool  OnClickCaption()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnClickButtonClose

Le gestionnaire d'évènement "ClickButtonClose" du contrôle.

```
virtual bool  OnClickButtonClose()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ClientAreaVisible

Définit une valeur indiquant si la zone client est visible ou pas.

```
bool  ClientAreaVisible(
   const bool  visible      // flag de visibilité
   )
```

Paramètres

visible

[in] Flag de visibilité.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ClientAreaLeft

Retourne la coordonnée X du coin supérieur gauche de la zone client du contrôle.

```
int  ClientAreaLeft()
```

Valeur de retour

La coordonnée X du coin supérieur gauche de la zone client du contrôle.

© 2000-2015, MetaQuotes Software Corp.

ClientAreaTop

Retourne la coordonnée Y du coin supérieur gauche de la zone client du contrôle.

```
int  ClientAreaTop()
```

Valeur de retour

La coordonnée Y du coin supérieur gauche de la zone client du contrôle.

© 2000-2015, MetaQuotes Software Corp.

ClientAreaRight

Retourne la coordonnée X du coin inférieur droit de la zone client du contrôle.

```
int  ClientAreaTop()
```

Valeur de retour

La coordonnée X du coin inférieur droit de la zone client du contrôle.

© 2000-2015, MetaQuotes Software Corp.

ClientAreaBottom

Retourne la coordonnée Y du coin inférieur droit de la zone client du contrôle.

```
int  ClientAreaBottom()
```

Valeur de retour

La coordonnée Y du coin inférieur droit de la zone client du contrôle.

© 2000-2015, MetaQuotes Software Corp.

ClientAreaWidth

Retourne la largeur de la zone client du contrôle.

```
int  ClientAreaWidth()
```

Valeur de retour

La largeur de la zone client du contrôle.

© 2000-2015, MetaQuotes Software Corp.

ClientAreaHeight

Retourne la hauteur de la zone client du contrôle.

```
int  ClientAreaHeight()
```

Valeur de retour

La hauteur de la zone client du contrôle.

© 2000-2015, MetaQuotes Software Corp.

OnDialogDragStart

Le gestionnaire d'évènement "DialogDragStart" du contrôle.

```
virtual bool  OnDialogDragStart()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DialogDragStart" est généré au démarrage du déplacement du contrôle (par Drag n' Drop).

© 2000-2015, MetaQuotes Software Corp.

OnDialogDragProcess

Le gestionnaire d'évènement "DialogDragProcess" du contrôle.

```
virtual bool  OnDialogDragProcess()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DialogDragProcess" est généré lorsque le contrôle est déplacé (par Drag n' Drop).

© 2000-2015, MetaQuotes Software Corp.

OnDialogDragEnd

Le gestionnaire d'évènement "DialogDragEnd" du contrôle.

```
virtual bool  OnDialogDragEnd()
```

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon.

Note

L'évènement "DialogDragEnd" est généré à la fin du déplacement du contrôle (par Drag n' Drop).

© 2000-2015, MetaQuotes Software Corp.

CAppDialog

La classe CAppDialog est une classe de contrôle complexe de Dialogue d'Application (avec des contrôles inter-dépendants).

Description

La classe CAppDialog permet de combiner des contrôles ayant des fonctions différentes dans un groupe au sein d'un programme MQL5.

Déclaration

```
class CAppDialog : public CDialog
```

Titre

```
#include <Controls\Dialog.mqh>
```

Méthodes de Classe

Création et destruction	
Create	Crée le contrôle
Destroy	Détruit le contrôle
Traitement des évènements	
OnEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Exécution	
Run	Exécute le contrôle
Traitement des évènements graphiques	
ChartEvent	Gestionnaire d'évènement à la base de tous les évènements graphiques
Paramètres	
Minimized	Définit une valeur indiquant si le contrôle est minimisé
Sauvegarde/Chargement	
IniFileSave	Sauvegarde l'état du contrôle dans un fichier
IniFileLoad	Charge l'état du contrôle depuis un fichier
IniFileName	Définit le nom du fichier de chargement/ sauvegarde de l'état du contrôle
IniFileExt	Définit l'extension du nom de fichier de chargement/sauvegarde de l'état du contrôle
Initialisation	

© 2000-2015, MetaQuotes Software Corp.

CreateCommon	Méthode standard d'initialisation
CreateExpert	Méthode d'initialisation pour travailler dans un Expert Advisor
CreateIndicator	Méthode d'initialisation pour travailler dans un indicateur
Contrôles dépendants	
CreateButtonMinMax	Crée des des contrôles dépendants (boutons minimiser et maximiser)
Gestionnaires d'évènements des contrôles dépendants	
OnClickButtonClose	Gestionnaire (virtuel) de l'évènement "ClickButtonClose"
OnClickButtonMinMax	Gestionnaire (virtuel) de l'évènement "ClickButtonMinMax"
Evènements externes	
OnAnotherApplicationClose	Gestionnaire (virtuel) d'évènements externes
Méthodes	
Rebound	Définit les nouvelles coordonnées du contrôle en utilisant les coordonnées de la classe CRect
Minimize	Réduit le contrôle
Maximize	Agrandit le contrôle à son état précédent
CreateInstanceId	Crée un identifiant unique pour les noms des objets de contrôle
ProgramName	Retourne le nom du programme MQL5
SubwinOff	Retourne le décalage Y de la sous-fenêtre du contrôle

© 2000-2015, MetaQuotes Software Corp.

Create

Crée un nouveau contrôle CAppDialog.

```
virtual bool  Create(
   const long     chart,      // identifiant du graphique
   const string   name,       // nom
   const int      subwin,     // sous-fenêtre du graphique
   const int      x1,         // coordonnée x1
   const int      y1,         // coordonnée y1
   const int      x2,         // coordonnée x2
   const int      y2          // coordonnée y2
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Destroy

Détruit le contrôle.

```
virtual bool  Destroy()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

OnEvent

Gestionnaire d'évènement graphique.

```
virtual bool  OnEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Run

Exécute le contrôle.

```
bool  Run()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

ChartEvent

Gestionnaire d'évènement graphique.

```
virtual bool  ChartEvent(
   const int      id,          // Identifiant
   const long&    lparam,      // paramètre d'évènement de type long
   const double&  dparam,      // paramètre d'évènement de type double
   const string&  sparam       // paramètre d'évènement de type string
   )
```

Paramètres

id

[in] Identifiant d'évènement.

lparam

[in] Paramètre d'évènement de type long, passé par référence.

dparam

[in] Paramètre d'évènement de type double, passé par référence.

sparam

[in] Paramètre d'évènement de type string, passé par référence.

Valeur de retour

vrai - si l'évènement a bien été traité, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Minimized

Définit la valeur de la propriété "Minimized" du contrôle.

```
bool  Minimized(
   const bool  flag      // état
   )
```

Paramètres

flag

[in] Nouvel état.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

IniFileSave

Sauvegarde l'état du contrôle dans un fichier

```
void  IniFileSave()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

IniFileLoad

Charge l'état du contrôle depuis un fichier

```
void  IniFileLoad()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

IniFileName

Définit le nom du fichier de chargement/sauvegarde de l'état du contrôle

```
virtual string  IniFileName()  const
```

Valeur de retour

Nom du fichier de chargement/sauvegarde de l'état du contrôle.

Note

Le nom du fichier inclut le nom de l'Expert Advisor/indicateur et le nom du symbole, sur lequel le programme MQL5 est lancé.

© 2000-2015, MetaQuotes Software Corp.

IniFileExt

Définit l'extension du nom de fichier de chargement/sauvegarde de l'état du contrôle

```
virtual string  IniFileExt()  const
```

Valeur de retour

Extension du fichier, utilisé pour le chargement/la sauvegarde de l'état du contrôle.

© 2000-2015, MetaQuotes Software Corp.

CreateCommon

Méthode standard d'initialisation.

```
bool  CreateCommon(
   const long    chart,      // identifiant du graphique
   const string  name,       // nom
   const int     subwin,     // sous-fenêtre du graphique
   )
```

Paramètres

chart

[in] Identifiant du graphique.

name

[in] Nom unique du contrôle.

subwin

[in] Sous-fenêtre du graphique.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateExpert

Méthode d'initialisation pour travailler dans un Expert Advisor

```
bool  CreateExpert(
   const int      x1,          // coordonnée x1
   const int      y1,          // coordonnée y1
   const int      x2,          // coordonnée x2
   const int      y2           // coordonnée y2
   )
```

Paramètres

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateIndicator

Méthode d'initialisation pour travailler dans un indicateur

```
bool  CreateIndicator(
   const int      x1,          // coordonnée x1
   const int      y1,          // coordonnée y1
   const int      x2,          // coordonnée x2
   const int      y2           // coordonnée y2
   )
```

Paramètres

x1

[in] Coordonnée X du coin supérieur gauche.

y1

[in] Coordonnée Y du coin supérieur gauche.

x2

[in] Coordonnée X du coin inférieur droit.

y2

[in] Coordonnée Y du coin inférieur droit.

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateButtonMinMax

Crée des contrôles inter-dépendants (boutons minimiser/maximiser).

```
virtual void  CreateButtonMinMax()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

OnClickButtonClose

Le gestionnaire d'évènement "ClickButtonClose" (clic de souris sur le bouton Fermer) du contrôle.

```
virtual void  OnClickButtonClose()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

OnClickButtonMinMax

Le gestionnaire d'évènement "ClickButtonMinMax" (clic de souris sur l'un des boutons minimiser/maximiser) du contrôle.

```
virtual void  OnClickButtonClose()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

OnAnotherApplicationClose

Gestionnaire d'évènement des évènements externes.

```
virtual void  OnAnotherApplicationClose()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Rebound

Définit les nouvelles coordonnées du contrôle utilisant les coordonnées de la classe CRect.

```
bool  Rebound(
   const & CRect  rect      // Classe CRect
   )
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Minimize

Réduit le contrôle

```
virtual void  Minimize()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

Maximize

Agrandit le contrôle à son état précédent.

```
virtual void  Maximize()
```

Valeur de retour

vrai - en cas de succès, faux sinon

© 2000-2015, MetaQuotes Software Corp.

CreateInstanceId

Crée un identifiant unique pour les noms des objets de contrôle.

```
string  CreateInstanceId()
```

Valeur de retour

Préfixe à utiliser pour les noms des objets.

© 2000-2015, MetaQuotes Software Corp.

ProgramName

Retourne le nom du programme MQL5.

```
string  ProgramName()
```

Valeur de retour

Nom du programme MQL5.

© 2000-2015, MetaQuotes Software Corp.

SubwinOff

Retourne le décalage Y de la sous-fenêtre du contrôle.

```
void  SubwinOff()
```

Valeur de retour

Aucune.

© 2000-2015, MetaQuotes Software Corp.

Le passage de MQL4

Le langage MQL5 est le développement du prédécesseur – le langage MQL4, dans laquel de nombreux indicateurs, des scripts et des experts ont été écrits. Malgré que le langage moderne de programmation est au maximum compatible avec le langage de la génération précédente, néanmoins il y a une série de différences entre ces langages. Et il faut connaître ces différences au transfert des programmes.

Dans ce paragraphe on a rassemblé l'information, qui est appelée à faciliter l'adaptation de ses codes à ces programmeurs qui savent bien MQL4, au langage nouveau MQL5.

Tout d'abord il est nécessaire de noter:

- Les fonctions start(), init() et deinit() manquent;
- Le nombre de tampons d'indicateur n'est pas limitée;
- Le chargement dll se passe à la fois après le chargement de l'expert (ou n'importe quel autre programme mql5);
- La vérification raccourcie des conditions logiques;
- À la sortie au-delà du tableau l'exécution courante se cesse (critiquement - avec la sortie de l'erreur);
- La priorité des opérations comme dans C++;
- La réduction implicite des types travaille (même de la chaîne dans le nombre);
- Les variables locales ne sont pas initialisées automatiquement (excepté les chaînes);
- Les tableaux ordinaires locaux sont supprimés automatiquement.

Les fonctions spéciales init, start et deinit

Dans le langage MQL4 était seulement trois fonctions prédéterminées, qui pouvaient être dans le code de l'indicateur, le conseiller ou le script (sans prendre en compte les fichiers inclus *.mqh et les fichiers des bibliothèques). Ces fonctions sont absentes à MQL5, mais il y a leurs analogues. Dans le tableau on présente la conformité approximative des fonctions.

MQL4	MQL5
init	OnInit
start	OnStart
deinit	OnDeinit

Les fonctions OnInit et OnDeinit jouent le même rôle que les fonctions init et deinit dans MQL4 – elles se sont destinées au placement du code, qui doit être accompli à l'initialisation et la déinitialisation des programmes mql5. Vous pouvez tout simplement rebaptiser ces fonctions de la bonne façon, ou les laisser comme elles sont, mais ajouter l'appel de ces fonctions dans les espaces correspondants.

Exemple:

© 2000-2015, MetaQuotes Software Corp.

```
void OnInit()
  {
//--- appelons la fonction à l'initialisation
   init();
  }
void OnDeinit(const int reason)
  {
//--- appelons la fonction à la déinitialisation
   deinit();
//---
  }
```

La fonction start est remplacée sur OnStart seulement dans les scripts, mais pour l'expert et l'indicateur il est nécessaire de la rebaptiser conformément à OnTick et OnCalculate. Notamment dans ces trois fonctions il est nécessaire d'installer le code, qui doit être accompli au temps de travail du programme mql5:

programme mql5	fonction principale
le script	OnStart
l'indicateur	OnCalculate
l'expert	OnTick

Si la fonction principale manque dans le code exécuté de l'indicateur ou le script, ou le nom de cette fonction se distingue de demandé, l'appel d'une telle fonction n'est pas produit. C'est-à-dire, si dans le code source du script il n'y a pas de fonction OnStart, un tel code sera compilé comme le conseiller.

Si dans le code de l'indicateur il n'y a pas de fonction OnCalculate, la compilation d'un tel indicateur est impossible.

Les variables prédéterminées

A MQL5 il n'y a pas de telles variables prédéterminées comme Ask, Bid, Bars. Les variables Digits et Point ont changé un peu dans l'orthographe, comme c'est montré dans le tableau.

MQL4	MQL5
Digits	_Digits
Point	_Point
	_LastError
	_Period
	_Symbol
	_StopFlag
	_UninitReason

L'accès aux séries temporelles

© 2000-2015, MetaQuotes Software Corp.

A MQL5 il n'y a pas des séries temporelles prédéterminées Open[], High[], Low[], Close[], Volume[] et Time[]. On peut spécifier la profondeur nécessaire de la série temporelle indépendamment à l'aide des fonctions correspondantes pour l'accès aux séries temporelles.

Les conseillers (les experts)

Les experts dans MQL5 doivent avoir pas absolument la fonction-gestionnaire de l'événement de l'entrée du nouveau tick OnTick, comme c'était à MQL4 (la fonction start dans MQL4 est appellée pour l'exécution à l'entrée du nouveau tick), parce que maintenant les experts dans MQL5 peuvent contenir les fonctions-gestionnaire de quelques types des événements:

- OnTick – L'entrée du nouveau tick;
- OnTimer –l'événement du minuteur;
- OnTrade - l'événement commercial;
- OnChartEvent – Les événements de l'entrée du clavier et de la souris, les événements de la navigation de l'objet graphique, l'événement de la fin de l'édition du texte dans la zone de l'entréé de l'objet LabelEdit;
- OnBookEvent – l'événement du changement de l'état du profondeur de marché (Depth of Market).

Les indicateurs d'utilisateur

A MQL4 le nombre de tampons d'indicateur est limitée et ne peut pas être plus que 8. Il n'y a pas de tel bornage dans MQL5, mais il faut se rappeler que chaque tampon d'indicateur demande la sélection du volume défini de la mémoire vive sous son placement dans le terminal, c'est pourquoi il ne faut pas abuser d'une nouvelle possibilité ouverte quand même.

En outre, dans MQL4 il y avait seulement 6 types du dessin de l'indicateur d'utilisateur, dans MQL5 maintenant il y a 18 styles du dessin. Et bien que les noms des aspects du dessin n'ont pas changé, l'idéologie de l'image graphique des indicateurs a changé beaucoup.

Se distingue également la direction de l'indexation dans les tampons d'indicateur. Dans MQL5 tous les tampons se comportent par défaut comme les tableaux ordinaires, c'est-à-dire l'élément avec l'index 0 est plus vieil à l'histoire, à l'augmentation de l'index nous nous avançons des données les plus vieilles vers les données dernières.

La seule fonction du travail avec les indicateurs d'utilisateur, qui a été préservé de MQL4 – c'est SetIndexBuffer. Mais son appel a changé aussi, maintenant il faut indiquer le type de données, qui se trouveront dans le tableau , lié au tampon d'indicateur.

En outre ont changé et se sont élargies les propriétés des indicateurs d'utilisateur, il y avait des fonctions de l'accès aux séries temporelles, c'est pourquoi il est nécessaire entièrement de repenser tout l'algorithme de calcul.

Les objets graphiques

Le nombre d'objets graphiques à MQL5 s'est élargie beaucoup. En outre le positionnement des objets graphiques dans le temps est devenu possible avec l'exactitude d'une seconde dans un graphique de n'importe quelle période – maintenant on ne produit pas l'arrondissement des points d'attachement des objets graphiques avec l'exactitude jusqu'au temps de l'ouverture de la barre sur le graphique du prix courant.

© 2000-2015, MetaQuotes Software Corp.

Pour les objets Arrow, Text et Label il y avait une possibilité d'indiquer le moyen du rattachement, et pour les objets Label, Button, Chart, Bitmap Label et Edit il y a la possibilité de spécifier l'angle du graphique, auquel l'objet est attaché.

© 2000-2015, MetaQuotes Software Corp.

List of MQL5 Functions

All MQL5 functions in alphabetical order.

Fonction	Action	Раздел
AccountInfoDouble	Rend la valeur du type double de la propriété correspondante du compte	Information sur le compte
AccountInfoInteger	Rend la valeur du type entier (bool,int ou long) de la propriété correspondante du compte	Information sur le compte
AccountInfoString	Rend la valeur du type string de la propriété correspondante du compte	Information sur le compte
acos	Rend la valeur d'arc cosinus x aux radians	Fonctions mathématiques
Alert	Affiche le message dans la fenêtre séparée	Fonctions communes
ArrayBsearch	Ищет указанное значение в отсортированном по возрастанию многомерном числовом массиве	Opérations avec les tableaux
ArrayCompare	Rend le résultat de la comparaison de deux tableux des types simples ou des structures d'utilisateur qui n'ont pas des objets complexes	Opérations avec les tableaux
ArrayCopy	Copie un tableau à l'autre	Opérations avec les tableaux
ArrayFill	Remplit le tableau numérique par la valeur indiquée	Opérations avec les tableaux
ArrayFree	Libère la mémoire du tampon de n'importe quel tableau dynamique et met la dimension de la valeur zéro au 0.	Opérations avec les tableaux
ArrayGetAsSeries	Contrôle la direction de l'indexation du tableau	Opérations avec les tableaux
ArrayInitialize	Met tous les éléments du tableau numérique à une valeur	Opérations avec les tableaux
ArrayIsDynamic	Vérifies - si le tableau est dynamique	Opérations avec les tableaux

© 2000-2015, MetaQuotes Software Corp.

ArrayIsSeries	Vérifies - si le tableau est la série temporelle	Opérations avec les tableaux
ArrayMaximum	Ищет максимальный элемент в первом измерении многомерного числового массива	Opérations avec les tableaux
ArrayMinimum	Ищет минимальный элемент в первом измерении многомерного числового массива	Opérations avec les tableaux
ArrayRange	Rend le nombre d'éléments dans la dimension indiquée du tableau	Opérations avec les tableaux
ArrayResize	Met la nouvelle grandeur dans la première dimension du tableau	Opérations avec les tableaux
ArraySetAsSeries	Établit la direction de l'indexation dans le tableau	Opérations avec les tableaux
ArraySize	Rend le nombre d'éléments dans le tableau	Opérations avec les tableaux
ArraySort	Сортирует многомерный числовой массив по возрастанию значений в первом измерении	Opérations avec les tableaux
asin	Rend la valeur d'arc sinus x aux radians	Fonctions mathématiques
atan	Rend l'arc tangent x aux radians	Fonctions mathématiques
Barres	Rend le nombre de barres aux histoires selon le symbole correspondant et la période	Accès aux timeséries et les données des indicateurs
BarsCalculated	Rend la quantité de données calculées dans le tampon d'indicateur ou-1 en cas de l'erreur (les données ne sont pas encore calculées)	Accès aux timeséries et les données des indicateurs
ceil	Rend la valeur entière numérique la plus proche par dessus	Fonctions mathématiques
CharArrayToString	Convertit le code de symbole (ansi) à la chaîne d'un symbole	Conversion des données
ChartApplyTemplate	Applique au graphique indiqué	Opérations avec les

© 2000-2015, MetaQuotes Software Corp.

	le modèle du fichier indiqué	graphiques
ChartClose	Ferme le graphique indiqué	Opérations avec les graphiques
ChartFirst	Rend l'identificateur du premier graphique du terminal de client	Opérations avec les graphiques
ChartGetDouble	Rend la valeur de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartGetInteger	Rend la valeur entière de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartGetString	Rend la valeur de chaîne de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartID	Rend l'identificateur du graphique courant	Opérations avec les graphiques
ChartIndicatorAdd	Ajoute l'indicateur avec le handle indiqué sur la fenêtre indiquée du graphique	Opérations avec les graphiques
ChartIndicatorDelete	Supprime l'indicateur avec le nom spécifié de la fenêtre indiquée du graphique	Opérations avec les graphiques
ChartIndicatorGet	Rend le handle de l'indicateur avec le nom court indiqué sur la fenêtre indiquée du graphique	Opérations avec les graphiques
ChartIndicatorName	Rend le nom court de l'indicateur selon le numéro dans la liste des indicateurs sur la fenêtre indiquée du graphique	Opérations avec les graphiques
ChartIndicatorsTotal	Rend le nombre de tous indicateurs qui sont attachés à la fenêtre indiquée du graphique	Opérations avec les graphiques
ChartNavigate	Réalise le décalage du graphique indiqué au nombre indiqué de barres par rapport à la position indiquée du graphique	Opérations avec les graphiques
ChartNext	Rend l'identificateur du graphique, suivant de	Opérations avec les graphiques

© 2000-2015, MetaQuotes Software Corp.

	l'indiqué	
ChartOpen	Ouvre un nouveau graphique avec le symbole indiqué et la période	Opérations avec les graphiques
CharToString	La conversion du code de symbole dans une chaîne d'un symbole	Conversion des données
ChartPeriod	Rend la valeur de la période du graphique indiqué	Opérations avec les graphiques
ChartPriceOnDropped	Rend la coordonnée de prix correspondant au point, dans laquelle on jette par la souris l'expert donné ou le script	Opérations avec les graphiques
ChartRedraw	Appelle la copie forcée du graphique indiqué	Opérations avec les graphiques
ChartSaveTemplate	Enregistre les paramètres actuels du graphique dans un modèle avec le nom spécifié	Opérations avec les graphiques
ChartScreenShot	Fournit un screenshot du graphique de son état actuel dans un format de GIF, PNG ou BMP en fonction de l'extension spécifiée	Opérations avec les graphiques
ChartSetDouble	Spécifie la valeur du type double de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartSetInteger	Spécifie la valeur du type entier (datetime, int, color, bool ou char) de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartSetString	Spécifie la valeur du type string de la propriété correspondante du graphique indiqué	Opérations avec les graphiques
ChartSetSymbolPeriod	Change les valeurs du symbole et la période du graphique indiqué	Opérations avec les graphiques
ChartSymbol	Rend le nom du symbole du graphique indiqué	Opérations avec les graphiques
ChartTimeOnDropped	Rend la coordonnée temporaire correspondant au	Opérations avec les graphiques

© 2000-2015, MetaQuotes Software Corp.

	point, dans laquelle on jette par la souris l'expert donné ou le script	
ChartTimePriceToXY	Convertit les coordonnées du graphique de la représentation graphique du temps/prix pour les axes X et Y	Opérations avec les graphiques
ChartWindowFind	Rend le numéro de sous-fenêtre, dans laquelle il y a un indicateur	Opérations avec les graphiques
ChartWindowOnDropped	Rend le numéro de sous-fenêtre du graphique, sur laquelle on jette l'expert donné, le script, l'objet ou l'indicateur par la souris	Opérations avec les graphiques
ChartXOnDropped	Rend la coordonnée selon l'axe X, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris	Opérations avec les graphiques
ChartXYToTimePrice	Convertit les coordonnées X et Y du graphique en valeurs - le temps et le prix	Opérations avec les graphiques
ChartYOnDropped	Rend la coordonnée selon l'axe Y, correspondant au point, dans lequel l'expert donné ou le script a été tombé par la souris	Opérations avec les graphiques
CheckPointer	Rend le type de l'indication de l'objet	Fonctions communes
CLBufferCreate	Crée le tampon OpenCL	Travail avec OpenCL
CLBufferFree	Supprime le tampon OpenCL	Travail avec OpenCL
CLBufferRead	Lit le tampon OpenCL au tableau	Travail avec OpenCL
CLBufferWrite	Inscrit le tableau au tampon OpenCL	Travail avec OpenCL
CLContextCreate	Crée le contexte OpenCL	Travail avec OpenCL
CLContextFree	Supprime le contexte OpenCL	Travail avec OpenCL
CLExecute	Exécute le programme OpenCL	Travail avec OpenCL
CLGetDeviceInfo	Reçoit la propriété du dispositif du pilote informatique OpenCL	Travail avec OpenCL

© 2000-2015, MetaQuotes Software Corp.

CLGetInfoInteger	Rend la valeur de la propriété entière pour l'objet OpenCL ou le dispositif	Travail avec OpenCL
CLHandleType	Rend le type OpenCL du handle à titre de la valeur de l'énumération ENUM_OPENCL_HANDLE_TYPE	Travail avec OpenCL
CLKernelCreate	Crée la fonction du lancement de OpenCL	Travail avec OpenCL
CLKernelFree	Supprime la fonction du lancement de OpenCL	Travail avec OpenCL
CLProgramCreate	Crée le programme OpenCL du code initial	Travail avec OpenCL
CLProgramFree	Supprime le programme OpenCL	Travail avec OpenCL
CLSetKernelArg	Expose le paramètre de la fonction OpenCL	Travail avec OpenCL
CLSetKernelArgMem	Expose le tampon OpenCL comme un paramètre de la fonction OpenCL	Travail avec OpenCL
ColorToARGB	Convertit le type color en type uint pour recevoir la représentation ARGB de la couleur.	Conversion des données
ColorToString	Convertit la valeur de la couleur à la chaîne du type "R,G,B"	Conversion des données
Comment	Affiche le message à un angle gauche supérieur du graphique de prix	Fonctions communes
CopyBuffer	Reçoit au tableau les données du tampon indiqué de l'indicateur indiqué	Accès aux timeséries et les données des indicateurs
CopyClose	Reçoit au tableau les données historiques au prix de la clôture des barres selon le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopyHigh	Reçoit au tableau les données historiques selon le prix maximum des barres selon le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs

© 2000-2015, MetaQuotes Software Corp.

CopyLow	Reçoit au tableau les données historiques selon le prix minimum des barres selon le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopyOpen	Reçoit au tableau les données historiques au prix de l'ouverture des barres selon le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopyRates	Reçoit au tableau les données historiques de la structure Rates pour le symbole et la période indiqués	Accès aux timeséries et les données des indicateurs
CopyRealVolume	Reçoit au tableau les données historiques par les volumes commerciaux pour le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopySpread	Reçoit au tableau les données historiques selon les spread pour le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopyTicks	Получает в массив тики, накопленные терминалом за текущую рабочую сессию	Accès aux timeséries et les données des indicateurs
CopyTickVolume	Reçoit au tableau les données historiques par les volumes des ticks pour le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
CopyTime	Reçoit au tableau les données historiques par le temps de l'ouverture des barres selon le symbole et la période correspondants	Accès aux timeséries et les données des indicateurs
cos	Rend le cosinus du nombre	Fonctions mathématiques
CryptDecode	Производит обратное преобразование данных массива	Fonctions communes
CryptEncode	Преобразует данные массива-источника в массив-приемник указанным методом	Fonctions communes
DebugBreak	Le point d'arrêt de programme de débogage	Fonctions communes

© 2000-2015, MetaQuotes Software Corp.

Digits	Rend le nombre de signes décimaux après la virgule, définissant l'exactitude de la mesure du prix du symbole du graphique courant	Vérification de l'état
DoubleToString	La conversion de la valeur numérique à la chaîne de texte avec l'exactitude indiquée	Conversion des données
EnumToString	La conversion de la valeur d'un transfert de n'importe quel type dans une chaîne	Conversion des données
EventChartCustom	Génère l'événement d'utilisateur pour le graphique indiqué	Travail avec les événements
EventKillTimer	Arrête sur le graphique courant la génération des événements selon le minuteur	Travail avec les événements
EventSetMillisecondTimer	Lance le générateur des événements de la minuterie d'une haute permission avec la période moins 1 seconde pour un graphique courant	Travail avec les événements
EventSetTimer	Lance le générateur des événements du minuteur avec la périodicité indiquée pour le graphique courant	Travail avec les événements
exp	Rend l'exposant au nombre	Fonctions mathématiques
ExpertRemove	Arrête le travail de l'expert et le décharge du graphique	Fonctions communes
fabs	Rend la valeur absolue (la valeur selon le module) du nombre transmis à elle	Fonctions mathématiques
FileClose	Ferme le fichier auparavant ouvert	Opérations de fichier
FileCopy	Copie le fichier original d'un répertoire local ou commun à un autre fichier	Opérations de fichier
FileDelete	Supprime le fichier indiqué	Opérations de fichier
FileFindClose	Ferme le handle de la recherche	Opérations de fichier
FileFindFirst	Commence l'excédent des fichiers dans le répertoire	Opérations de fichier

© 2000-2015, MetaQuotes Software Corp.

	correspondant conformément au filtre indiqué	
FileFindNext	Continue la recherche commencée par la fonction FileFindFirst()	Opérations de fichier
FileFlush	Écrit à un disque toutes les données restant dans le tampon du fichier d'entrée-de sortie	Opérations de fichier
FileGetInteger	Reçoit la propriété entière du fichier	Opérations de fichier
FileIsEnding	Définit la fin du fichier en train de la lecture	Opérations de fichier
FileIsExist	Vérifie l'existence d'un fichier	Opérations de fichier
FileIsLineEnding	Définit la fin de la ligne dans le fichier du texte en train de la lecture	Opérations de fichier
FileMove	Déplace et renomme le fichier	Opérations de fichier
FileOpen	Ouvre un fichier avec un nom et un drapeau indiqués	Opérations de fichier
FileReadArray	Lit les tableaux de tous les types, sauf les tableaux de chaîne (peut être un tableau des structures qui ne contient pas les chaînes et les tableaux dynamiques), du fichier binaire de la position courante de l'indicateur de fichier	Opérations de fichier
FileReadBool	Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type bool	Opérations de fichier
FileReadDatetime	Lit du fichier du type CSV la chaîne d'un des formats: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" ou "HH:MI:SS" - et la transforme en valeur du type datetime	Opérations de fichier
FileReadDouble	Lit le nombre de l'exactitude double avec une virgule flottante (double) du fichier binaire de la position courante	Opérations de fichier

© 2000-2015, MetaQuotes Software Corp.

	de l'indicateur de fichier	
FileReadFloat	Lit de la position courante de l'indicateur de fichier la valeur du type float	Opérations de fichier
FileReadInteger	Lit du fichier binaire la valeur du type int, short ou char en fonction de la longueur indiquée dans les bytes	Opérations de fichier
FileReadLong	Lit de la position courante de l'indicateur de fichier la valeur du type long	Opérations de fichier
FileReadNumber	Lit du fichier du type CSV la chaîne de la position courante jusqu'au délimiteur (ou jusqu'à la fin de la ligne de texte) et transforme la chaîne lue en valeur du type double	Opérations de fichier
FileReadString	Lit du fichier la chaîne de la position courante de l'indicateur de fichier	Opérations de fichier
FileReadStruct	Lit du fichier binaire le contenu à la structure transmise à titre du paramètre	Opérations de fichier
FileSeek	Déplace la position de l'indicateur de fichier par un nombre spécifié de bytes par rapport à la position indiquée	Opérations de fichier
FileSize	Rend la dimension d'un fichier ouvert correspondant	Opérations de fichier
FileTell	Rend la position courante de l'indicateur de fichier du fichier correspondant ouvert	Opérations de fichier
FileWrite	Inscrit les données au fichier comme CSV ou TXT	Opérations de fichier
FileWriteArray	Inscrit au fichier du type BIN les tableaux des tous les types, sauf les tableaux de chaîne	Opérations de fichier
FileWriteDouble	Inscrit au fichier binaire la valeur du paramètre du type double de la position courante de l'indicateur de fichier	Opérations de fichier
FileWriteFloat	Inscrit au fichier binaire la	Opérations de fichier

© 2000-2015, MetaQuotes Software Corp.

	valeur du paramètre du type float de la position courante de l'indicateur de fichier	
FileWriteInteger	Inscrit au fichier binaire la valeur du paramètre du type int de la position courante de l'indicateur de fichier	Opérations de fichier
FileWriteLong	Inscrit au fichier binaire la valeur du paramètre du type long de la position courante de l'indicateur de fichier	Opérations de fichier
FileWriteString	Inscrit au fichier du type BIN ou TXT la valeur du paramètre du type string de la position courante de l'indicateur de fichier	Opérations de fichier
FileWriteStruct	Inscrit au fichier binaire le contenu de la structure, transmise à titre du paramètre de la position courante de l'indicateur de fichier	Opérations de fichier
floor	Rend la valeur entière numérique la plus proche par dessous	Fonctions mathématiques
fmax	Rend la valeur maximal des deux valeurs numériques	Fonctions mathématiques
fmin	Rend la valeur minimal des deux valeurs numériques	Fonctions mathématiques
fmod	Rend le reste réel après la division de deux nombres	Fonctions mathématiques
FolderClean	Supprime tous les fichiers dans le dossier indiqué	Opérations de fichier
FolderCreate	Crée le répertoire dans le dossier Files (en fonction de la valeur common_flag)	Opérations de fichier
FolderDelete	Supprime le répertoire indiqué. Si le dossier n'est pas vide, elle ne peut pas être supprimé	Opérations de fichier
FrameAdd	Ajoute la trame avec les données	Travail avec les résultats de l'optimisation
FrameFilter	Définit le filtre de la lecture des trames et met l'indicateur	Travail avec les résultats de l'optimisation

© 2000-2015, MetaQuotes Software Corp.

	au début	
FrameFirst	Convertit l'indicateur de la lecture des trames au début et oblitére un filtre établi.	Travail avec les résultats de l'optimisation
FrameInputs	Reçoit les paramètres input, sur lesquels la trame a été formé	Travail avec les résultats de l'optimisation
FrameNext	Lit la trame et déplace l'indicateur au suivant	Travail avec les résultats de l'optimisation
GetLastError	Rend la valeur de la dernière erreur	Vérification de l'état
GetPointer	Rend l'indicateur de l'objet	Fonctions communes
GetTickCount	Rend la quantité des millisecondes qui ont passé dès le moment du départ du système	Fonctions communes
GlobalVariableCheck	Contrôle l'existence de la variable globale avec le nom indiqué	Variables globales du terminal de client
GlobalVariableDel	Supprime la variable globale	Variables globales du terminal de client
GlobalVariableGet	Demande la valeur de la variable globale	Variables globales du terminal de client
GlobalVariableName	Rend le nom de la variable globale selon le numéro d'ordre dans le répertoire des variables globales	Variables globales du terminal de client
GlobalVariablesDeleteAll	Supprime les variables globales avec le préfixe indiqué dans le nom	Variables globales du terminal de client
GlobalVariableSet	Met la nouvelle valeur à une variable globale	Variables globales du terminal de client
GlobalVariableSetOnCondition	Met une nouvelle valeur de la variable existant globale selon la condition	Variables globales du terminal de client
GlobalVariablesFlush	Inscrit forcément le contenu de toutes les variables globales sur le disque	Variables globales du terminal de client
GlobalVariablesTotal	Rend le total des variables globales	Variables globales du terminal de client
GlobalVariableTemp	Met une nouvelle valeur de la	Variables globales du terminal

© 2000-2015, MetaQuotes Software Corp.

	variable globale, qui existe seulement pendant la session courante du terminal	de client
GlobalVariableTime	Rend le temps du dernier accès à la variable globale	Variables globales du terminal de client
HistoryDealGetDouble	Rend la propriété demandée du marché dans l'histoire (double)	Fonctions commerciales
HistoryDealGetInteger	Rend la propriété demandée du marché dans l'histoire (datetime ou int)	Fonctions commerciales
HistoryDealGetString	Rend la propriété demandée du marché dans l'histoire (string)	Fonctions commerciales
HistoryDealGetTicket	Choisit le marché pour le traitement ultérieur et rend le ticket les marchés dans l'histoire	Fonctions commerciales
HistoryDealSelect	Choisit le marché dans l'histoire pour davantage l'appeler par les fonctions appropriées	Fonctions commerciales
HistoryDealsTotal	Rend le nombre de marchés dans l'histoire	Fonctions commerciales
HistoryOrderGetDouble	Rend la propriété demandée de l'ordre dans l'histoire (double)	Fonctions commerciales
HistoryOrderGetInteger	Rend la propriété demandée de l'ordre dans l'histoire (datetime ou int)	Fonctions commerciales
HistoryOrderGetString	Rend la propriété demandée de l'ordre dans l'histoire (string)	Fonctions commerciales
HistoryOrderGetTicket	Rend le ticket de l'ordre correspondant dans l'histoire	Fonctions commerciales
HistoryOrderSelect	Choisit à l'histoire l'ordre pour le travail ultérieur avec lui	Fonctions commerciales
HistoryOrdersTotal	Rend le nombre d'ordres dans l'histoire	Fonctions commerciales
HistorySelect	Demande l'histoire des marchés et des ordres pour la période indiquée du temps de	Fonctions commerciales

© 2000-2015, MetaQuotes Software Corp.

	serveur	
HistorySelectByPosition	Demande l'histoire des marchés et des ordres avecl'identificateur indiqué de la position.	Fonctions commerciales
iAC	Accelerator Oscillator	Indicateurs techniques
iAD	Accumulation/Distribution	Indicateurs techniques
iADX	Average Directional Index	Indicateurs techniques
iADXWilder	Average Directional Index by Welles Wilder	Indicateurs techniques
iAlligator	Alligator	Indicateurs techniques
iAMA	Adaptive Moving Average	Indicateurs techniques
iAO	Awesome Oscillator	Indicateurs techniques
iATR	Average True Range	Indicateurs techniques
iBands	Bollinger Bands®	Indicateurs techniques
iBearsPower	Bears Power	Indicateurs techniques
iBullsPower	Bulls Power	Indicateurs techniques
iBWMFI	Market Facilitation Index by Bill Williams	Indicateurs techniques
iCCI	Commodity Channel Index	Indicateurs techniques
iChaikin	Chaikin Oscillator	Indicateurs techniques
iCustom	l'indicateur d'utilisateur	Indicateurs techniques
iDEMA	Double Exponential Moving Average	Indicateurs techniques
iDeMarker	DeMarker	Indicateurs techniques
iEnvelopes	Envelopes	Indicateurs techniques
iForce	Force Index	Indicateurs techniques
iFractals	Fractals	Indicateurs techniques
iFrAMA	Fractal Adaptive Moving Average	Indicateurs techniques
iGatore	Gator Oscillator	Indicateurs techniques
iIchimoku	Ichimoku Kinko Hyo	Indicateurs techniques
iMA	Moving Average	Indicateurs techniques
iMACD	Moving Averages Convergence-Divergence	Indicateurs techniques

© 2000-2015, MetaQuotes Software Corp.

iMFI	Money Flow Index	Indicateurs techniques
iMomentum	Momentum	Indicateurs techniques
IndicatorCreate	Rend le handle de l'indicateur indiqué technique créé à la base du tableau des paramètres du type MqlParam	Accès aux timeséries et les données des indicateurs
IndicatorParameters	Rend le nombre de paramètres d'entrée de l'indicateur selon le handle indiqué, ainsi que les valeurs elles-mêmes et le type des paramètres	Accès aux timeséries et les données des indicateurs
IndicatorRelease	Supprime le handle de l'indicateur et libère la partie calculée de l'indicateur, si personne ne se sert plus d'elle	Accès aux timeséries et les données des indicateurs
IndicatorSetDouble	Spécifie la valeur de la propriété d'indicateur, ayant le type double	Indicateurs d'utilisateur
IndicatorSetInteger	Spécifie la valeur de la propriété d'indicateur, ayant le type int	Indicateurs d'utilisateur
IndicatorSetString	Spécifie la valeur de la propriété d'indicateur, ayant le type string	Indicateurs d'utilisateur
IntegerToString	La conversion de la valeur du type entier à la chaîne de la longueur indiquée	Conversion des données
iOBV	On Balance Volume	Indicateurs techniques
iOsMA	Moving Average of Oscillator (MACD histogram)	Indicateurs techniques
iRSI	Relative Strength Index	Indicateurs techniques
iRVI	Relative Vigor Index	Indicateurs techniques
iSAR	Parabolic Stop And Reverse System	Indicateurs techniques
IsStopped	Rend true, si on a donné l'ordre de terminer l'exécution du programme mql5	Vérification de l'état
iStdDev	Standard Deviation	Indicateurs techniques
iStochastic	Stochastic Oscillator	Indicateurs techniques
iTEMA	Triple Exponential Moving Average	Indicateurs techniques

© 2000-2015, MetaQuotes Software Corp.

iTriX	Triple Exponential Moving Averages Oscillator	Indicateurs techniques
iVIDyA	Variable Index DYnamic Average	Indicateurs techniques
iVolumes	Volumes	Indicateurs techniques
iWPR	Williams' Percent Range	Indicateurs techniques
log	Rend le logarithme naturel	Fonctions mathématiques
log10	Rend le logarithme du nombre selon le fondement 10	Fonctions mathématiques
MarketBookAdd	Assure l'ouverture du profondeur de marché selon l'instrument indiqué, ainsi que produit la souscription sur la réception des avis du changement du profondeur de marché indiqué	Réception de l'information de marché
MarketBookGet	Rend le tableau des structures du type MqlBookInfo, contenant les enregistrements du profondeur de marché du symbole indiqué	Réception de l'information de marché
MarketBookRelease	Assure la clôture du profondeur de marché selon l'instrument, ainsi qui élimine la souscription sur la réception des avis du changement du profondeur de marché indiqué	Réception de l'information de marché
MathAbs	Rend la valeur absolue (la valeur selon le module) du nombre transmis à elle	Fonctions mathématiques
MathArccos	Rend la valeur d'arc cosinus x aux radians	Fonctions mathématiques
MathArcsin	Rend la valeur d'arc sinus x aux radians	Fonctions mathématiques
MathArctan	Rend l'arc tangent x aux radians	Fonctions mathématiques
MathCeil	Rend la valeur entière numérique la plus proche par dessus	Fonctions mathématiques
MathCos	Rend le cosinus du nombre	Fonctions mathématiques
MathExp	Rend l'exposant au nombre	Fonctions mathématiques

© 2000-2015, MetaQuotes Software Corp.

MathFloor	Rend la valeur entière numérique la plus proche par dessous	Fonctions mathématiques
MathIsValidNumber	Vérifie l'exactitude d'un nombre réel	Fonctions mathématiques
MathLog	Rend le logarithme naturel	Fonctions mathématiques
MathLog10	Rend le logarithme du nombre selon le fondement 10	Fonctions mathématiques
MathMax	Rend la valeur maximal des deux valeurs numériques	Fonctions mathématiques
MathMin	Rend la valeur minimal des deux valeurs numériques	Fonctions mathématiques
MathMod	Rend le reste réel après la division de deux nombres	Fonctions mathématiques
MathPow	Elève le fondement à la puissance indiquée	Fonctions mathématiques
MathRand	Rend le nombre entier pseudo-accidentelle dans le domaine de 0 jusqu'à 32767	Fonctions mathématiques
MathRound	Arrondit le nombre jusqu'à l'entier plus proche	Fonctions mathématiques
MathSin	Rend le sinus du nombre	Fonctions mathématiques
MathSqrt	Rend la racine carrée	Fonctions mathématiques
MathSrand	Établit l'état initial du générateur des nombres entiers quasi-aléatoires	Fonctions mathématiques
MathTan	Rend la tangent du nombre	Fonctions mathématiques
MessageBox	Crée, affiche la fenêtre des messages et la dirige	Fonctions communes
MQLInfoInteger	Rend la valeur du type entier de la propriété correspondante du programme mql5 lancé	Vérification de l'état
MQLInfoString	Rend la valeur du type string de la propriété correspondante du programme mql5 lancé	Vérification de l'état
NormalizeDouble	L'arrondissement du nombre de la virgule flottante à l'exactitude indiquée	Conversion des données
ObjectCreate	Crée l'objet du type spécifié sur le graphique indiqué	Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

ObjectDelete	Supprime l'objet avec le nom indiqué du graphique indiqué (de la sous-fenêtre de graphique indiquée)	Objets graphiques
ObjectFind	Cherche l'objet selon le nom avec l'identificateur indiqué	Objets graphiques
ObjectGetDouble	Rend la valeur du type double de la propriété correspondante de l'objet	Objets graphiques
ObjectGetInteger	Rend la valeur entière de la propriété correspondante de l'objet	Objets graphiques
ObjectGetString	Rend la valeur du type string de la propriété correspondante de l'objet	Objets graphiques
ObjectGetTimeByValue	Rend la valeur du temps pour la valeur indiquée du prix de l'objet	Objets graphiques
ObjectGetValueByTime	Rend la valeur de prix de l'objet pour le temps indiqué	Objets graphiques
ObjectMove	Change les coordonnées du point indiqué du rattachement de l'objet	Objets graphiques
ObjectName	Rend le nom de l'objet du type correspondant dans le graphique indiqué (indiqué dans la sous- fenêtre du graphique)	Objets graphiques
ObjectsDeleteAll	Supprime tous les objets du type indiqué du graphique indiqué (de la sous-fenêtre de graphique indiquée)	Objets graphiques
ObjectSetDouble	Met la valeur de la propriété correspondante de l'objet	Objets graphiques
ObjectSetInteger	Met la valeur de la propriété correspondante de l'objet	Objets graphiques
ObjectSetString	Met la valeur de la propriété correspondante de l'objet	Objets graphiques
ObjectsTotal	Rend le nombre d'objets du type indiqué dans le graphique indiqué (la sous-fenêtre de graphique indiquée)	Objets graphiques

© 2000-2015, MetaQuotes Software Corp.

OrderCalcMargin	Calcule le montant de la marge nécessaire au type indiqué de l'ordre, en devise du compte	Fonctions commerciales
OrderCalcProfit	Calcule le montant du bénéfice en vertu des paramètres transmis en devise du compte	Fonctions commerciales
OrderCheck	Contrôle la suffisance des moyens pour l'accomplissement de la transaction commerciale demandée.	Fonctions commerciales
OrderGetDouble	Rend la propriété demandée de l'ordre (double)	Fonctions commerciales
OrderGetInteger	Rend la propriété demandée de l'ordre (datetime ou int)	Fonctions commerciales
OrderGetString	Rend la propriété demandée de l'ordre (string)	Fonctions commerciales
OrderGetTicket	Rend le ticket de l'ordre correspondant	Fonctions commerciales
OrderSelect	Choisit l'ordre pour le travail ultérieur avec lui	Fonctions commerciales
OrderSend	Expédie les demandes commerciales au serveur	Fonctions commerciales
OrderSendAsync	Envoie les demandes commerciales asynchronement sans attente de la réponse du serveur commercial	Fonctions commerciales
OrdersTotal	Rend le nombre d'ordres	Fonctions commerciales
ParameterGetRange	Reçoit pour la variable inputl'information sur la gamme des valeurs et sur le pas du changement à l'optimisation de l'expert dans le testeur des stratégies	Travail avec les résultats de l'optimisation
ParameterSetRange	Établit des règles de l'utilisation de la variable inputà l'optimisation de l'expert dans le testeur des stratégies: la valeur, le pas du changement, les valeurs initiale et finale	Travail avec les résultats de l'optimisation
Period	Rend la valeur du temps trame du graphique courant	Vérification de l'état

© 2000-2015, MetaQuotes Software Corp.

PeriodSeconds	Rend la quantité de secondes dans la période	Fonctions communes
PlaySound	Reproduit le fichier sonore	Fonctions communes
PlotIndexGetInteger	Rend la valeur de la propriété de la ligne d'indicateur, ayant le type entier	Indicateurs d'utilisateur
PlotIndexSetDouble	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type double	Indicateurs d'utilisateur
PlotIndexSetInteger	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type int	Indicateurs d'utilisateur
PlotIndexSetString	Spécifie la valeur de la propriété de la ligne d'indicateur, ayant le type string	Indicateurs d'utilisateur
Point	Rend la dimension de point de l'instrument courant en devise de la cotation	Vérification de l'état
PositionGetDouble	Rend la propriété demandée de la position ouverte (double)	Fonctions commerciales
PositionGetInteger	Rend la propriété demandée de la position ouverte (datetime ou int)	Fonctions commerciales
PositionGetString	Rend la propriété demandée de la position ouverte (string)	Fonctions commerciales
PositionGetSymbol	Rend le symbole de la position correspondante ouverte	Fonctions commerciales
PositionSelect	Choisit la position ouverte pour le travail ultérieur avec elle	Fonctions commerciales
PositionsTotal	Rend le nombre de positions ouvertes	Fonctions commerciales
pow	Elève le fondement à la puissance indiquée	Fonctions mathématiques
Print	Affiche un message dans le journal	Fonctions communes
PrintFormat	Formate et imprime l'ensemble des symboles et des valeurs dans un fichier-journal conformément au format	Fonctions communes

© 2000-2015, MetaQuotes Software Corp.

	donné	
rand	Rend le nombre entier pseudo-accidentelle dans le domaine de 0 jusqu'à 32767	Fonctions mathématiques
ResetLastError	Établit la valeur de la variable prédéterminée _LastError au zéro	Fonctions communes
ResourceCreate	Crée la ressource de l'image à la base de l'ensemble des données	Fonctions communes
ResourceFree	Supprime la ressource dynamiquement créée (libère la mémoire occupée par la ressource)	Fonctions communes
ResourceReadImage	Lit les données de la ressource graphique, créé par la fonction ResourceCreate() ou sauvegardée dans le fichier EX5 à la compilation	Fonctions communes
ResourceSave	Sauvegarde la ressource au fichier spécifié	Fonctions communes
round	Arrondit le nombre jusqu'à l'entier plus proche	Fonctions mathématiques
SendFTP	Envoie le fichier à l'adresse indiquée dans la fenêtre des réglages du signet "la publication"	Fonctions communes
SendMail	Envoie le message électronique à l'adresse indiquée dans la fenêtre des réglages du signet "la Poste"	Fonctions communes
SendNotification	Envoie les notifications Push aux terminaux mobiles, dont les ID MetaQuotes sont indiqués sur l'onglet "Notifications"	Fonctions communes
SeriesInfoInteger	Rend l'information sur l'état des données historiques	Accès aux timeséries et les données des indicateurs
SetIndexBuffer	Attache le tampon d'indicateur indiquée avec le tableau unidimentionnel dynamique du type double	Indicateurs d'utilisateur
ShortArrayToString	Copie la partie du tableau dans	Conversion des données

© 2000-2015, MetaQuotes Software Corp.

	la chaîne	
ShortToString	La conversion du code de symbole (unicode)à la chaîne d'un symbole	Conversion des données
SignalBaseGetDouble	Возвращает значение свойства типа double для выбранного сигнала	Trade Signals
SignalBaseGetInteger	Возвращает значение свойства типа integer для выбранного сигнала	Trade Signals
SignalBaseGetString	Возвращает значение свойства типа string для выбранного сигнала	Trade Signals
SignalBaseSelect	Выбирает для работы сигнал из базы торговых сигналов, доступных в терминале	Trade Signals
SignalBaseTotal	Возвращает общее количество сигналов, доступных в терминале	Trade Signals
SignalInfoGetDouble	Возвращает из настроек копирования торгового сигнала значение свойства типа double	Trade Signals
SignalInfoGetInteger	Возвращает из настроек копирования торгового сигнала значение свойства типа integer	Trade Signals
SignalInfoGetString	Возвращает из настроек копирования торгового сигнала значение свойства типа string	Trade Signals
SignalInfoSetDouble	Устанавливает в настройках копирования торгового сигнала значение свойства типа double	Trade Signals
SignalInfoSetInteger	Устанавливает в настройках копирования торгового сигнала значение свойства типа integer	Trade Signals
SignalSubscribe	Производит подписку на копирование торгового сигнала	Trade Signals
SignalUnsubscribe	Отменяет подписку на	Trade Signals

© 2000-2015, MetaQuotes Software Corp.

	копирование торгового сигнала	
sin	Rend le sinus du nombre	Fonctions mathématiques
Sleep	Retient l'exécution de l'expert courant ou le script sur l'intervalle défini	Fonctions communes
sqrt	Rend la racine carrée	Fonctions mathématiques
srand	Établit l'état initial du générateur des nombres entiers quasi-aléatoires	Fonctions mathématiques
StringAdd	Ajoute la chaîne selon la place indiquée par la sous-chaîne	Fonctions de chaîne
StringBufferLen	Rend la grandeur du buffer distribué pour la chaîne.	Fonctions de chaîne
StringCompare	Compare deux chaînes et rend 1 si la première chaîne est plus que deuxième; 0 - si les chaînes sont égales; -1 (moins un) - si la première chaîne est moins que la deuxième	Fonctions de chaîne
StringConcatenate	Forme la chaîne des paramètres transmis	Fonctions de chaîne
StringFill	Remplit la chaîne indiquée selon la place par les symboles indiqués	Fonctions de chaîne
StringFind	La recherche de la sous-chaîne dans la chaîne	Fonctions de chaîne
StringFormat	Convertit le nombre à la chaîne conformément au format spécifié	Conversion des données
StringGetCharacter	Rend la valeur du symbole disposé dans la position indiquée de la chaîne	Fonctions de chaîne
StringInit	Initialise la chaîne par les symboles indiqués et assure la grandeur indiquée de la chaîne	Fonctions de chaîne
StringLen	Rend le nombre de symboles dans la chaîne	Fonctions de chaîne
StringReplace	Remplace tous les sous-chaînes recherchés dans la chaîne pour la séquence spécifiée des caractères.	Fonctions de chaîne

© 2000-2015, MetaQuotes Software Corp.

StringSetCharacter	Rend la copie de la chaîne avec la valeur changée du symbole dans la position indiquée	Fonctions de chaîne
StringSplit	Reçoit les sous-chaînes de la chaîne indiquée par un séparateur spécifié et rend le montant des sous-chaînes reçus	Fonctions de chaîne
StringSubstr	Extrait la sous-chaîne de la chaîne de texte qui commence de la position indiquée	Fonctions de chaîne
StringToCharArray	Copie par un symbole la chaîne convertie de Unicode à ANSI à l'endroit spécifié du tableau du type uchar	Conversion des données
StringToColor	Convertit la chaîne du type "R,G,B" ou la chaîne, contenant le nom de la couleur, à la valeur du type color	Conversion des données
StringToDouble	La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type double	Conversion des données
StringToInteger	La conversion de la chaîne, contenant la représentation symbolique du nombre, dans le nombre du type int	Conversion des données
StringToLower	Transforme tous les symboles de la chaîne indiquée aux symboles minuscules selon la place	Fonctions de chaîne
StringToShortArray	Copie la chaîne par un symbole à l'endroit spécifié du tableau du type ushort	Conversion des données
StringToTime	La conversion de la chaîne, contenant le temps et/ou la date dans le format "yyyy.mm.dd [hh:mi]", dans le nombre du type datetime	Conversion des données
StringToUpper	Transforme tous les symboles de la chaîne indiquée aux symboles majuscules selon la place	Fonctions de chaîne

© 2000-2015, MetaQuotes Software Corp.

StringTrimLeft	Supprime les symboles de la conversion de chariot, les espaces et les symboles de la tabulation du début de la chaîne	Fonctions de chaîne
StringTrimRight	Supprime les symboles de la conversion de chariot, les espaces et les symboles de la tabulation à la fin de la chaîne	Fonctions de chaîne
StructToTime	Produit la conversion de la valeur du type de la structure MqlDateTime à la variable du type datetime	Date et temps
Symbol	Rend le nom du symbole du graphique courant	Vérification de l'état
SymbolInfoDouble	Rend la valeur du type double du symbole indiqué pour la propriété correspondante	Réception de l'information de marché
SymbolInfoInteger	Rend la valeur du type entier (long, datetime, int ou bool) du symbole indiqué pour la propriété correspondante	Réception de l'information de marché
SymbolInfoMarginRate	Возвращает коэффициенты взимания маржи в зависимости от типа и направления ордера	Réception de l'information de marché
SymbolInfoSessionQuote	Permet de recevoir le temps du début et le temps de la fin de la session de cotation indiquée pour le symbole et le jour de la semaine indiqués.	Réception de l'information de marché
SymbolInfoSessionTrade	Permet de recevoir le temps du début et le temps de la fin de la session commerciale indiquée pour le symbole et le jour de la semaine indiqués.	Réception de l'information de marché
SymbolInfoString	Rend la valeur du type string du symbole indiqué pour la propriété correspondante	Réception de l'information de marché
SymbolInfoTick	Rend les valeurs courantes pour le symbole indiqué dans la variable du type MqlTick	Réception de l'information de marché
SymbolIsSynchronized	Vérifie le synchronismedes données selon le symbole	Réception de l'information de marché

© 2000-2015, MetaQuotes Software Corp.

	indiqué dans le terminal avec les données sur le serveur commercial	
SymbolName	Rend le nom du symbole indiqué	Réception de l'information de marché
SymbolSelect	Choisit le symbole dans la fenêtre MarketWatch ou enlève le symbole de la fenêtre	Réception de l'information de marché
SymbolsTotal	Rend le nombre accessible (choisi dans MarketWatch ou tous)des symboles	Réception de l'information de marché
tan	Rend la tangent du nombre	Fonctions mathématiques
TerminalClose	Envoie au terminal l'ordre sur l'achèvement du travail	Fonctions communes
TerminalInfoDouble	Rend la valeur du type double de la propriété correspondante de l'environnement du programme mql5	Vérification de l'état
TerminalInfoInteger	Rend la valeur du type entier de la propriété correspondante de l'environnement du programme mql5	Vérification de l'état
TerminalInfoString	Rend la valeur du type string de la propriété correspondante de l'environnement du programme mql5	Vérification de l'état
TesterStatistics	Rend la valeur du paramètre indiqué statistique, calculée d'après les résultats du test	Fonctions communes
TextGetSize	Rend la largeur et la hauteur de la ligne aux réglages courants de la fonte	Objets graphiques
TextOut	Déduit le texte au tableau d'utilisateur (le tampon) destiné à la création de la ressourcegraphique	Objets graphiques
TextSetFont	Établit la fonte pour la sortie du texte par les méthodes du dessin (on utilise par défaut la fonte Arial 20)	Objets graphiques
TimeCurrent	Rend le dernier temps connu du serveur (le temps de l'arrivée de la dernière	Date et temps

© 2000-2015, MetaQuotes Software Corp.

	cotation) dans le format datetime	
TimeDaylightSavings	Rend le signe du passage sur l'heure d'été/l'hiver	Date et temps
TimeGMT	Rend le temps GMT au format datetime en tenant compte du passage sur l'heure d'hiver ou d'été par le temps local de l'ordinateur, sur lequel le terminal de client a été lancé	Date et temps
TimeGMTOffset	Rend la différence actuelle entre le temps GMT et le temps local de l'ordinateur en secondes en tenant compte du passage sur l'heure d'hiver ou d'été	Date et temps
TimeLocal	Rend le temps local informatique dans le format datetime	Date et temps
TimeToString	La conversion de la valeur, contenant le temps en secondes, passé après le 01.01.1970, à la chaîne du format "yyyy.mm.dd hh:mi"	Conversion des données
TimeToStruct	Produit la conversion de la valeur du type datetime à la variable du type de la structure MqlDateTime	Date et temps
TimeTradeServer	Rend le temps calculé courant du serveur commercial	Date et temps
UninitializeReason	Rend le code de la raison de la déinitialisation	Vérification de l'état
WebRequest	Отправляет HTTP-запрос на указанный сервер	Fonctions communes
ZeroMemory	Remet au zéro la variable transmise selon la référence. La variable peut être de n'importe quel type, seulement les classes et les structures qui ont les constructeurs font l'exception	Fonctions communes

© 2000-2015, MetaQuotes Software Corp.

List of MQL5 Constants

All MQL5 constants in alphabetical order.

Constante	Description	Использование
__DATE__	La date de la compilation du fichier sans l'heure (les heures, les minutes et les secondes sont égales 0)	Print
__DATETIME__	La date et le temps de la compilation du fichier	Print
__FILE__	Le nom du fichier courant compilé	Print
__FUNCSIG__	La signature de la fonction, dont dans le corps la macro est située. La sortie de la description complète de la fonction avec les types des paramètres dans le log peut être utile à l'identification des fonctions surchargées	Print
__FUNCTION__	Le nom de la fonction, dans le corps duquel se trouve la macro	Print
__LINE__	Le numéro de la ligne dans le code source, où se trouve la macro donnée	Print
__MQLBUILD__, __MQL5BUILD__	Le numéro du build du compilateur	Print
__PATH__	Le chemin absolu vers le fichier compilé actuel	Print
ACCOUNT_ASSETS	Текущий размер активов на счёте	AccountInfoDouble
ACCOUNT_BALANCE	La balance du compte en devise du dépôt	AccountInfoDouble
ACCOUNT_COMMISSION_BLOCKED	Текущая сумма заблокированных комиссий по счёту	AccountInfoDouble
ACCOUNT_COMPANY	Le nom d'une compagnie qui sert le compte	AccountInfoString
ACCOUNT_CREDIT	Le montant du crédit accordé en devise du dépôt	AccountInfoDouble

© 2000-2015, MetaQuotes Software Corp.

ACCOUNT_CURRENCY	La devise du dépôt	AccountInfoString
ACCOUNT_EQUITY	La valeur des moyens propres sur le compte en devise du dépôt	AccountInfoDouble
ACCOUNT_LEVERAGE	Le montant de l'épaule accordée	AccountInfoInteger
ACCOUNT_LIABILITIES	Текущий размер обязательств на счёте	AccountInfoDouble
ACCOUNT_LIMIT_ORDERS	La quantité au maximum admissible d'ordres actifs remis	AccountInfoInteger
ACCOUNT_LOGIN	Le numéro de compte	AccountInfoInteger
ACCOUNT_MARGIN	Le montant des moyens réservés hypothécaires sur le compte en devise du dépôt	AccountInfoDouble
ACCOUNT_MARGIN_FREE	Le montant des moyens libres sur le compte en devise du dépôt, accessible pour l'ouverture de la position	AccountInfoDouble
ACCOUNT_MARGIN_INITIAL	Размер средств, зарезервированных на счёте, для обеспечения гарантийной суммы по всем отложенным ордерам	AccountInfoDouble
ACCOUNT_MARGIN_LEVEL	Le niveau des moyens hypothécaires sur le compte en pourcentage	AccountInfoDouble
ACCOUNT_MARGIN_MAINTENANCE	Размер средств, зарезервированных на счёте, для обеспечения минимальной суммы по всем открытым позициям	AccountInfoDouble
ACCOUNT_MARGIN_SO_CALL	Margin call level. Depending on the set ACCOUNT_MARGIN_SO_MODE is expressed in percents or in the deposit currency. ACCOUNT_MARGIN_SO_MODE s'exprime en pourcentage ou en devise du dépôt	AccountInfoDouble
ACCOUNT_MARGIN_SO_MODE	Le mode de la spécification du niveau minimum admissible des moyens hypothécaires	AccountInfoInteger

© 2000-2015, MetaQuotes Software Corp.

ACCOUNT_MARGIN_SO_SO	Margin stop out level. Depending on the set ACCOUNT_MARGIN_SO_MODE is expressed in percents or in the deposit currency. ACCOUNT_MARGIN_SO_MODE s'exprime en pourcentage ou en devise du dépôt	AccountInfoDouble
ACCOUNT_NAME	Le nom du client	AccountInfoString
ACCOUNT_PROFIT	Le montant du bénéfice courant sur le compte en devise du dépôt	AccountInfoDouble
ACCOUNT_SERVER	Le nom du serveur commercial	AccountInfoString
ACCOUNT_STOPOUT_MODE_MONEY	Le niveau est donné dans l'argent	AccountInfoInteger
ACCOUNT_STOPOUT_MODE_PERCENT	Le niveau est donné en pourcentage	AccountInfoInteger
ACCOUNT_TRADE_ALLOWED	Le commerce permis pour le compte courant	AccountInfoInteger
ACCOUNT_TRADE_EXPERT	Le commerce permis pour l'expert	AccountInfoInteger
ACCOUNT_TRADE_MODE	Account trade mode	AccountInfoInteger
ACCOUNT_TRADE_MODE_CONTEST	Le compte de concours commercial	AccountInfoInteger
ACCOUNT_TRADE_MODE_DEMO	Le compte commercial de démonstration	AccountInfoInteger
ACCOUNT_TRADE_MODE_REAL	Le compte réel commercial	AccountInfoInteger
ALIGN_CENTER	L'alignement selon le centre (seulement pourl'objet "Champ de l'entrée")	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_LEFT	L'alignement selon une frontière gauche	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_RIGHT	L'alignement selon une frontière droite	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ANCHOR_CENTER	Le point du rattachement strictement selon le centre de l'objet	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT	Le point du rattachement à gauche selon le centre	ObjectSetInteger, ObjectGetInteger

© 2000-2015, MetaQuotes Software Corp.

ANCHOR_LEFT_LOWER	Le point du rattachement dans un angle gauche inférieur	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_UPPER	Le point du rattachement dans un angle gauche supérieur	ObjectSetInteger, ObjectGetInteger
ANCHOR_LOWER	Le point du rattachement d'en bas selon le centre	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT	Le point du rattachement à droite selon le centre	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_LOWER	Le point du rattachement dans un angle droit inférieur	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_UPPER	Le point du rattachement à droite selon le centre	ObjectSetInteger, ObjectGetInteger
ANCHOR_UPPER	Le point du rattachement par dessus selon le centre	ObjectSetInteger, ObjectGetInteger
BASE_LINE	Une ligne principale	Lignes des indicateurs
BOOK_TYPE_BUY	La demande pour l'achat	MqlBookInfo
BOOK_TYPE_BUY_MARKET	La demande pour l'achat au prix du marché	MqlBookInfo
BOOK_TYPE_SELL	La demande pour la vente	MqlBookInfo
BOOK_TYPE_SELL_MARKET	La demande pour la vente au prix du marché	MqlBookInfo
BORDER_FLAT	Forme plate	ObjectSetInteger, ObjectGetInteger
BORDER_RAISED	Forme proéminente	ObjectSetInteger, ObjectGetInteger
BORDER_SUNKEN	Forme concave	ObjectSetInteger, ObjectGetInteger
CHAR_MAX	La valeur maximale, qui peut être présentée par le type char	Constantes des types de données numériques
CHAR_MIN	La valeur minimale, qui peut être présentée par le type char	Constantes des types de données numériques
CHART_AUTOSCROLL	Le mode du déplacement automatique vers le bord droit du graphique	ChartSetInteger, ChartGetInteger
CHART_BARS	L'affichage comme les barres	ChartSetInteger
CHART_BEGIN	Le début du graphique (les prix les plus vieux)	ChartNavigate
CHART_BRING_TO_TOP	L'affichage du graphique au-dessus de tous les autres	ChartSetInteger, ChartGetInteger

© 2000-2015, MetaQuotes Software Corp.

CHART_CANDLES	L'affichage comme les chandeliers japonais	ChartSetInteger
CHART_COLOR_ASK	La couleur de la ligne du prix Ask	ChartSetInteger, ChartGetInteger
CHART_COLOR_BACKGROUND	La couleur du fond du graphique	ChartSetInteger, ChartGetInteger
CHART_COLOR_BID	La couleur de la ligne du prix Bid	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BEAR	La couleur du corps de la bougie de baissier	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BULL	La couleur du corps de la bougie d'haussier	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_DOWN	La couleur de la barre en bas, de l'ombre et de l'encadrement du corps de la bougie de baissier	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_LINE	La couleur de la ligne du graphique et des chandeliers japonais "Doji"	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_UP	La couleur de la barre en haut, de l'ombre et de l'encadrement du corps de la bougie d'haussier	ChartSetInteger, ChartGetInteger
CHART_COLOR_FOREGROUND	La couleur des axes, les échelles et les lignes OHLC	ChartSetInteger, ChartGetInteger
CHART_COLOR_GRID	La couleur de la grille	ChartSetInteger, ChartGetInteger
CHART_COLOR_LAST	La couleur de la ligne du prix du dernier marché passé (Last)	ChartSetInteger, ChartGetInteger
CHART_COLOR_STOP_LEVEL	La couleur des niveaux des ordres Stop (Stop Loss et Take Profit)	ChartSetInteger, ChartGetInteger
CHART_COLOR_VOLUME	La couleur des volumes et des niveaux de l'ouverture des positions	ChartSetInteger, ChartGetInteger
CHART_COMMENT	Le texte du commentaire sur le graphique	ChartSetString, ChartGetString
CHART_CURRENT_POS	La position courante	ChartNavigate
CHART_DRAG_TRADE_LEVELS	L'autorisation de faire glisser les niveaux commerciaux sur	ChartSetInteger, ChartGetInteger

© 2000-2015, MetaQuotes Software Corp.

	le graphique à l'aide de la souris. Par défaut, le mode glisser-déposer est activé (la valeur true)	
CHART_EVENT_MOUSE_MOVE	L'expédition des messages à tous les programmes mql5 sur le graphique sur les événements du déplacement et les pressions des boutons de la souris (CHARTEVENT_MOUSE_MOVE)	ChartSetInteger, ChartGetInteger
CHART_EVENT_OBJECT_CREATE	L'expédition des messages à tous les programmes mql5 sur le graphique sur l'événement de la création de l'objet graphique (CHARTEVENT_OBJECT_CREATE)	ChartSetInteger, ChartGetInteger
CHART_EVENT_OBJECT_DELETE	L'expédition des messages à tous les programmes mql5 sur le graphique sur l'événement de la destruction de l'objet graphique (CHARTEVENT_OBJECT_DELETE)	ChartSetInteger, ChartGetInteger
CHART_FIRST_VISIBLE_BAR	Le numéro de la première barre visible sur le graphique. L'indexation des barres correspond à la série temporelle.	ChartSetInteger, ChartGetInteger
CHART_FIXED_MAX	Le maximum fixé du graphique	ChartSetDouble, ChartGetDouble
CHART_FIXED_MIN	Le minimum fixé du graphique	ChartSetDouble, ChartGetDouble
CHART_FIXED_POSITION	La situation de la position fixée du graphique du bout gauche en pourcentage. La position fixée du graphique est désignée par un petit triangle gris sur l'axe horizontal du temps et est montrée seulement dans le cas où on déconnecte le défilement automatique vers le bord droit à l'entrée du nouveau tick (regarde la	ChartSetDouble, ChartGetDouble

© 2000-2015, MetaQuotes Software Corp.

	propriété CHART_AUTOSCROLL). La barre, qui se trouve sur la position fixée, reste à la même place à l'augmentation et la réduction de l'échelle.	
CHART_FOREGROUND	Le graphique des prix dans l'arrière-plan	ChartSetInteger, ChartGetInteger
CHART_HEIGHT_IN_PIXELS	La hauteur du graphique en pixels	ChartSetInteger, ChartGetInteger
CHART_IS_OBJECT	Le critère pour l'identification de l'objet "Graphique" (OBJ_CHART) - rend true pour l'objet graphique. Pour un vrai graphique la valeur est égale au false	ChartSetInteger, ChartGetInteger
CHART_LINE	L'affichage comme une ligne tracée par les prix Close	ChartSetInteger
CHART_MODE	Le type de graphique (les bougies, les barres ou la ligne)	ChartSetInteger, ChartGetInteger
CHART_MOUSE_SCROLL	Le défilement du graphique par un bouton gauche de la souris à l'horizontale. Le défilement selon la verticale sera aussi accessible, si dans true est établi une valeur de l'une des trois propriétés: CHART_SCALEFIX, CHART_SCALEFIX_11 ou CHART_SCALE_PT_PER_BAR	ChartSetInteger, ChartGetInteger
CHART_POINTS_PER_BAR	La signification de l'échelle dans les points sur la barre	ChartSetDouble, ChartGetDouble
CHART_PRICE_MAX	Le maximum du graphique	ChartSetDouble, ChartGetDouble
CHART_PRICE_MIN	Le minimum du graphique	ChartSetDouble, ChartGetDouble
CHART_SCALE	Echelle	ChartSetInteger, ChartGetInteger
CHART_SCALE_PT_PER_BAR	Le mode de l'indication de l'échelle dans les points sur la barre	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX	Le mode de l'échelle fixée	ChartSetInteger, ChartGetInteger

© 2000-2015, MetaQuotes Software Corp.

CHART_SCALEFIX_11	Le mode de l'échelle 1:1	ChartSetInteger, ChartGetInteger
CHART_SHIFT	Le mode de l'alinéa du bord droit du graphique de prix	ChartSetInteger, ChartGetInteger
CHART_SHIFT_SIZE	Le montant de l'alinéa de la barre nulle du bord droit en pourcentage	ChartSetDouble, ChartGetDouble
CHART_SHOW_ASK_LINE	L'affichage de la signification Ask par la ligne horizontale sur le graphique	ChartSetInteger, ChartGetInteger
CHART_SHOW_BID_LINE	L'affichage de la signification Bid par la ligne horizontale sur le graphique	ChartSetInteger, ChartGetInteger
CHART_SHOW_DATE_SCALE	L'affichage de l'échelle du temps sur le graphique	ChartSetInteger, ChartGetInteger
CHART_SHOW_GRID	L'affichage de la grille dans le graphique	ChartSetInteger, ChartGetInteger
CHART_SHOW_LAST_LINE	L'affichage de la signification Last par la ligne horizontale sur le graphique	ChartSetInteger, ChartGetInteger
CHART_SHOW_OBJECT_DESCR	Les descriptions surgissantes des objets graphiques	ChartSetInteger, ChartGetInteger
CHART_SHOW_OHLC	L'affichage dans un gauche angle supérieur des significations OHLC	ChartSetInteger, ChartGetInteger
CHART_SHOW_ONE_CLICK	Отображение на графике панели быстрой торговли (опция "Торговля одним кликом")	ChartSetInteger, ChartGetInteger
CHART_SHOW_PERIOD_SEP	L'affichage des délimiteurs verticaux entre les périodes voisines	ChartSetInteger, ChartGetInteger
CHART_SHOW_PRICE_SCALE	L'affichage de l'échelle de prix sur le graphique	ChartSetInteger, ChartGetInteger
CHART_SHOW_TRADE_LEVELS	L'affichage sur le graphique des niveaux commerciaux (les niveaux des positions ouvertes, Stop Loss, Take Profit et des ordres remis)	ChartSetInteger, ChartGetInteger
CHART_SHOW_VOLUMES	L'affichage des volumes sur le graphique	ChartSetInteger, ChartGetInteger
CHART_VISIBLE_BARS	Le nombre de barres sur le	ChartSetInteger,

© 2000-2015, MetaQuotes Software Corp.

	graphique, accessible pour l'affichage	ChartGetInteger
CHART_VOLUME_HIDE	Les volumes ne sont pas montrés	ChartSetInteger
CHART_VOLUME_REAL	Les volumes de commerce	ChartSetInteger
CHART_VOLUME_TICK	Les volumes de tick	ChartSetInteger
CHART_WIDTH_IN_BARS	La largeur du graphique dans les barres	ChartSetInteger, ChartGetInteger
CHART_WIDTH_IN_PIXELS	La largeur du graphique en pixels	ChartSetInteger, ChartGetInteger
CHART_WINDOW_HANDLE	Le handle de graphique (HWND)	ChartSetInteger, ChartGetInteger
CHART_WINDOW_IS_VISIBLE	La visibilité des sous - fenêtres	ChartSetInteger, ChartGetInteger
CHART_WINDOW_YDISTANCE	La distance en pixels selon l'axe vertical Y entre le cadre supérieur de la sous-fenêtre de l'indicateur et le cadre supérieur de la fenêtre principale du graphique. A l'arrivée des événements de la souris les coordonnées du curseur sont transmises aux coordonnées de la fenêtre principale du graphique, pendant que les coordonnées des objets graphiques dans une sous-fenêtre de l'indicateur sont spécifiées relativement un angle gauche supérieur de la sous-fenêtre. La valeur est demandée pour la transfert des coordonnées absolues du graphique principal aux coordonnées locales de la sous-fenêtre pour le travail correct avec les objets graphiques, dont les coordonnées sont spécifiées relativement un angle gauche supérieur du cadre de la sous-fenêtre.	ChartSetInteger, ChartGetInteger
CHART_WINDOWS_TOTAL	Le total des fenêtres du graphique, y compris les sous - fenêtres des indicateurs	ChartSetInteger, ChartGetInteger

© 2000-2015, MetaQuotes Software Corp.

CHARTEVENT_CHART_CHANGE	Le changement des dimensions du graphique ou le changement des propriétés du graphique par le dialogue des propriétés	OnChartEvent
CHARTEVENT_CLICK	Cliquer sur le graphique	OnChartEvent
CHARTEVENT_CUSTOM	Le numéro initial de l'événement du domaine des événements d'utilisateur	OnChartEvent
CHARTEVENT_CUSTOM_LAST	Le numéro final de l'événement du domaine des événements d'utilisateur	OnChartEvent
CHARTEVENT_KEYDOWN	Préssion de clavier	OnChartEvent
CHARTEVENT_MOUSE_MOVE	Перемещение мыши и нажатие кнопок мыши (если для графика установлено свойство CHART_EVENT_MOUSE_MOVE= true)	OnChartEvent
CHARTEVENT_OBJECT_CHANGE	L'événement du changement des propriétés de l'objet graphique par le dialogue des propriétés	OnChartEvent
CHARTEVENT_OBJECT_CLICK	Cliquer sur un objet graphique	OnChartEvent
CHARTEVENT_OBJECT_CREATE	La création de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_DELETE=true)	OnChartEvent
CHARTEVENT_OBJECT_DELETE	L'effacement de l'objet graphique (si la propriété est définie pour le graphique CHART_EVENT_OBJECT_DELETE=true)	OnChartEvent
CHARTEVENT_OBJECT_DRAG	Le glisser-déposer de l'objet graphique	OnChartEvent
CHARTEVENT_OBJECT_ENDEDIT	La fin de l'édition du texte dans l'objet graphique Edit	OnChartEvent
CHARTS_MAX	Le nombre maximale possible des graphiques ouverts ensemble dans le terminal	Les autres constantes
CHIKOUSPAN_LINE	La ligne Chikou Span	Lignes des indicateurs

© 2000-2015, MetaQuotes Software Corp.

clrAliceBlue	Alice Blue	Ensemble des couleurs Web
clrAntiqueWhite	Antique White	Ensemble des couleurs Web
clrAqua	Aqua	Ensemble des couleurs Web
clrAquamarine	Aquamarine	Ensemble des couleurs Web
clrBeige	Beige	Ensemble des couleurs Web
clrBisque	Bisque	Ensemble des couleurs Web
clrBlack	Black	Ensemble des couleurs Web
clrBlanchedAlmond	Blanched Almond	Ensemble des couleurs Web
clrBlue	Blue	Ensemble des couleurs Web
clrBlueViolet	Blue Violet	Ensemble des couleurs Web
clrBrown	Brown	Ensemble des couleurs Web
clrBurlyWood	Burly Wood	Ensemble des couleurs Web
clrCadetBlue	Cadet Blue	Ensemble des couleurs Web
clrChartreuse	Chartreuse	Ensemble des couleurs Web
clrChocolate	Chocolate	Ensemble des couleurs Web
clrCoral	Coral	Ensemble des couleurs Web
clrCornflowerBlue	Cornflower Blue	Ensemble des couleurs Web
clrCornsilk	Cornsilk	Ensemble des couleurs Web
clrCrimson	Crimson	Ensemble des couleurs Web
clrDarkBlue	Dark Blue	Ensemble des couleurs Web
clrDarkGoldenrod	Dark Goldenrod	Ensemble des couleurs Web
clrDarkGray	Dark Gray	Ensemble des couleurs Web
clrDarkGreen	Dark Green	Ensemble des couleurs Web
clrDarkKhaki	Dark Khaki	Ensemble des couleurs Web
clrDarkOliveGreen	Dark Olive Green	Ensemble des couleurs Web
clrDarkOrange	Dark Orange	Ensemble des couleurs Web
clrDarkOrchid	Dark Orchid	Ensemble des couleurs Web
clrDarkSalmon	Dark Salmon	Ensemble des couleurs Web
clrDarkSeaGreen	Dark Sea Green	Ensemble des couleurs Web
clrDarkSlateBlue	Dark Slate Blue	Ensemble des couleurs Web
clrDarkSlateGray	Dark Slate Gray	Ensemble des couleurs Web
clrDarkTurquoise	Dark Turquoise	Ensemble des couleurs Web

© 2000-2015, MetaQuotes Software Corp.

clrDarkViolet	Dark Violet	Ensemble des couleurs Web
clrDeepPink	Deep Pink	Ensemble des couleurs Web
clrDeepSkyBlue	Deep Sky Blue	Ensemble des couleurs Web
clrDimGray	Dim Gray	Ensemble des couleurs Web
clrDodgerBlue	Dodger Blue	Ensemble des couleurs Web
clrFireBrick	Fire Brick	Ensemble des couleurs Web
clrForestGreen	Forest Green	Ensemble des couleurs Web
clrGainsboro	Gainsboro	Ensemble des couleurs Web
clrGold	Gold	Ensemble des couleurs Web
clrGoldenrod	Goldenrod	Ensemble des couleurs Web
clrGray	Gray	Ensemble des couleurs Web
clrGreen	Green	Ensemble des couleurs Web
clrGreenYellow	Green Yellow	Ensemble des couleurs Web
clrHoneydew	Honeydew	Ensemble des couleurs Web
clrHotPink	Hot Pink	Ensemble des couleurs Web
clrIndianRed	Indian Red	Ensemble des couleurs Web
clrIndigo	Indigo	Ensemble des couleurs Web
clrIvory	Ivory	Ensemble des couleurs Web
clrKhaki	Khaki	Ensemble des couleurs Web
clrLavender	Lavender	Ensemble des couleurs Web
clrLavenderBlush	Lavender Blush	Ensemble des couleurs Web
clrLawnGreen	Lawn Green	Ensemble des couleurs Web
clrLemonChiffon	Lemon Chiffon	Ensemble des couleurs Web
clrLightBlue	Light Blue	Ensemble des couleurs Web
clrLightCoral	Light Coral	Ensemble des couleurs Web
clrLightCyan	Light Cyan	Ensemble des couleurs Web
clrLightGoldenrod	Light Goldenrod	Ensemble des couleurs Web
clrLightGray	Light Gray	Ensemble des couleurs Web
clrLightGreen	Light Green	Ensemble des couleurs Web
clrLightPink	Light Pink	Ensemble des couleurs Web
clrLightSalmon	Light Salmon	Ensemble des couleurs Web
clrLightSeaGreen	Light Sea Green	Ensemble des couleurs Web

© 2000-2015, MetaQuotes Software Corp.

clrLightSkyBlue	Light Sky Blue	Ensemble des couleurs Web
clrLightSlateGray	Light Slate Gray	Ensemble des couleurs Web
clrLightSteelBlue	Light Steel Blue	Ensemble des couleurs Web
clrLightYellow	Light Yellow	Ensemble des couleurs Web
clrLime	Lime	Ensemble des couleurs Web
clrLimeGreen	Lime Green	Ensemble des couleurs Web
clrLinen	Linen	Ensemble des couleurs Web
clrMagenta	Magenta	Ensemble des couleurs Web
clrMaroon	Maroon	Ensemble des couleurs Web
clrMediumAquamarine	Medium Aquamarine	Ensemble des couleurs Web
clrMediumBlue	Medium Blue	Ensemble des couleurs Web
clrMediumOrchid	Medium Orchid	Ensemble des couleurs Web
clrMediumPurple	Medium Purple	Ensemble des couleurs Web
clrMediumSeaGreen	Medium Sea Green	Ensemble des couleurs Web
clrMediumSlateBlue	Medium Slate Blue	Ensemble des couleurs Web
clrMediumSpringGreen	Medium Spring Green	Ensemble des couleurs Web
clrMediumTurquoise	Medium Turquoise	Ensemble des couleurs Web
clrMediumVioletRed	Medium Violet Red	Ensemble des couleurs Web
clrMidnightBlue	Midnight Blue	Ensemble des couleurs Web
clrMintCream	Mint Cream	Ensemble des couleurs Web
clrMistyRose	Misty Rose	Ensemble des couleurs Web
clrMoccasin	Moccasin	Ensemble des couleurs Web
clrNavajoWhite	Navajo White	Ensemble des couleurs Web
clrNavy	Navy	Ensemble des couleurs Web
clrNONE	L'absence de la couleur	Les autres constantes
clrOldLace	Old Lace	Ensemble des couleurs Web
clrOlive	Olive	Ensemble des couleurs Web
clrOliveDrab	Olive Drab	Ensemble des couleurs Web
clrOrange	Orange	Ensemble des couleurs Web
clrOrangeRed	Orange Red	Ensemble des couleurs Web
clrOrchid	Orchid	Ensemble des couleurs Web
clrPaleGoldenrod	Pale Goldenrod	Ensemble des couleurs Web

© 2000-2015, MetaQuotes Software Corp.

clrPaleGreen	Pale Green	Ensemble des couleurs Web
clrPaleTurquoise	Pale Turquoise	Ensemble des couleurs Web
clrPaleVioletRed	Pale Violet Red	Ensemble des couleurs Web
clrPapayaWhip	Papaya Whip	Ensemble des couleurs Web
clrPeachPuff	Peach Puff	Ensemble des couleurs Web
clrPeru	Peru	Ensemble des couleurs Web
clrPink	Pink	Ensemble des couleurs Web
clrPlum	Plum	Ensemble des couleurs Web
clrPowderBlue	Powder Blue	Ensemble des couleurs Web
clrPurple	Purple	Ensemble des couleurs Web
clrRed	Red	Ensemble des couleurs Web
clrRosyBrown	Rosy Brown	Ensemble des couleurs Web
clrRoyalBlue	Royal Blue	Ensemble des couleurs Web
clrSaddleBrown	Saddle Brown	Ensemble des couleurs Web
clrSalmon	Salmon	Ensemble des couleurs Web
clrSandyBrown	Sandy Brown	Ensemble des couleurs Web
clrSeaGreen	Sea Green	Ensemble des couleurs Web
clrSeashell	Seashell	Ensemble des couleurs Web
clrSienna	Sienna	Ensemble des couleurs Web
clrSilver	Silver	Ensemble des couleurs Web
clrSkyBlue	Sky Blue	Ensemble des couleurs Web
clrSlateBlue	Slate Blue	Ensemble des couleurs Web
clrSlateGray	Slate Gray	Ensemble des couleurs Web
clrSnow	Snow	Ensemble des couleurs Web
clrSpringGreen	Spring Green	Ensemble des couleurs Web
clrSteelBlue	Steel Blue	Ensemble des couleurs Web
clrTan	Tan	Ensemble des couleurs Web
clrTeal	Teal	Ensemble des couleurs Web
clrThistle	Thistle	Ensemble des couleurs Web
clrTomato	Tomato	Ensemble des couleurs Web
clrTurquoise	Turquoise	Ensemble des couleurs Web
clrViolet	Violet	Ensemble des couleurs Web

© 2000-2015, MetaQuotes Software Corp.

clrWheat	Wheat	Ensemble des couleurs Web
clrWhite	White	Ensemble des couleurs Web
clrWhiteSmoke	White Smoke	Ensemble des couleurs Web
clrYellow	Yellow	Ensemble des couleurs Web
clrYellowGreen	Yellow Green	Ensemble des couleurs Web
CORNER_LEFT_LOWER	Le centre des coordonnées dans un angle gauche inférieur du graphique	ObjectSetInteger, ObjectGetInteger
CORNER_LEFT_UPPER	Le centre des coordonnées dans un angle gauche supérieur du graphique	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_LOWER	Le centre des coordonnées dans un angle droit inférieur du graphique	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_UPPER	Le centre des coordonnées dans un angle droit supérieur du graphique	ObjectSetInteger, ObjectGetInteger
CP_ACP	Une page de code courant ANSI le codage au système opérationnel Windows	CharArrayToString, StringToCharArray, FileOpen
CP_MACCP	Une page de code courant Macintosh. Note: Cette signification est utilisée principalement dans les codes de programme auparavant créés et maintenant on n'a pas besoin de lui, puisque les ordinateurs modernes Macintosh utilisent le codage Unicode.	CharArrayToString, StringToCharArray, FileOpen
CP_OEMCP	Une page de code courant OEM.	CharArrayToString, StringToCharArray, FileOpen
CP_SYMBOL	La page de code Symbol	CharArrayToString, StringToCharArray, FileOpen
CP_THREAD_ACP	Le codage Windows ANSI pour le thread courant d'exécution.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF7	La page de code UTF-7.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF8	La page de code UTF-8.	CharArrayToString, StringToCharArray, FileOpen
CRYPT_AES128	Шифрование AES с ключом	CryptEncode, CryptDecode

© 2000-2015, MetaQuotes Software Corp.

	128 бит (16 байт)	
CRYPT_AES256	Шифрование AES с ключом 256 бит (32 байта)	CryptEncode, CryptDecode
CRYPT_ARCH_ZIP	ZIP архивирование	CryptEncode, CryptDecode
CRYPT_BASE64	Шифрование BASE64 (перекодировка)	CryptEncode, CryptDecode
CRYPT_DES	Шифрование DES с ключом 56 бит (7 байт)	CryptEncode, CryptDecode
CRYPT_HASH_MD5	Расчёт HASH MD5	CryptEncode, CryptDecode
CRYPT_HASH_SHA1	Расчёт HASH SHA1	CryptEncode, CryptDecode
CRYPT_HASH_SHA256	Расчёт HASH SHA256	CryptEncode, CryptDecode
DBL_DIG	La quantité de signes signifiants décimaux	Constantes des types de données numériques
DBL_EPSILON	La valeur minimale, à qui satisfait la condition : 1.0 +DBL_EPSILON != 1.0	Constantes des types de données numériques
DBL_MANT_DIG	La quantité de bits dans la mantisse	Constantes des types de données numériques
DBL_MAX	La valeur maximale, qui peut être présentée par le type double	Constantes des types de données numériques
DBL_MAX_10_EXP	La valeur maximale décimale du degré d'exposant	Constantes des types de données numériques
DBL_MAX_EXP	La valeur maximale binaire du degré d'exposant	Constantes des types de données numériques
DBL_MIN	La valeur minimale positive, qui peut être présentée par le type double	Constantes des types de données numériques
DBL_MIN_10_EXP	La valeur minimale décimale du degré d'exposant	Constantes des types de données numériques
DBL_MIN_EXP	La valeur minimale binaire du degré d'exposant	Constantes des types de données numériques
DEAL_COMMENT	Le commentaire au le marché	HistoryDealGetString
DEAL_COMMISSION	La commission du marché	HistoryDealGetDouble
DEAL_ENTRY	La direction du marché – les entrées au marché, la sortie du marché ou le retournement	HistoryDealGetInteger
DEAL_ENTRY_IN	L'entrée au marché	HistoryDealGetInteger

© 2000-2015, MetaQuotes Software Corp.

DEAL_ENTRY_INOUT	L'inversion	HistoryDealGetInteger
DEAL_ENTRY_OUT	La sortie du marché	HistoryDealGetInteger
DEAL_MAGIC	Magic number pour la position (regardes ORDER_MAGIC)	HistoryDealGetInteger
DEAL_ORDER	L'ordre, à la base de quel le marché est effectué	HistoryDealGetInteger
DEAL_POSITION_ID	L'identificateur de la position, à l'ouverture, au changement ou à la clôture où ce marché a participé. Chaque position possède l'identificateur unique, qui est attribué à tous les marchés, exécutés à l'instrument pendant toute la durée de vie de la position.	HistoryDealGetInteger
DEAL_PRICE	Le prix du marché	HistoryDealGetDouble
DEAL_PROFIT	Le résultat financier du marché	HistoryDealGetDouble
DEAL_SWAP	Le swap accumulé à la clôture	HistoryDealGetDouble
DEAL_SYMBOL	Le nom du symbole, selon lequel le marché est produit	HistoryDealGetString
DEAL_TIME	Le temps du marché	HistoryDealGetInteger
DEAL_TIME_MSC	Le temps de l'accomplissement du marché en millisecondes depuis le 01.01.1970	HistoryDealGetInteger
DEAL_TYPE	Le type du marché	HistoryDealGetInteger
DEAL_TYPE_BALANCE	La mise en compte de la balance	HistoryDealGetInteger
DEAL_TYPE_BONUS	L'énumération des bonus	HistoryDealGetInteger
DEAL_TYPE_BUY	L'achat	HistoryDealGetInteger
DEAL_TYPE_BUY_CANCELED	Le marché annulé de l'achat. La situation est possible, quand le marché sur l'achat auparavant fait est annulé. Dans ce cas, le type du marché auparavant fait (DEAL_TYPE_BUY) se change sur DEAL_TYPE_BUY_CANCELED, et son bénéfice/perte est remise à zéro. Le bénéfice/perte auparavant reçu est	HistoryDealGetInteger

© 2000-2015, MetaQuotes Software Corp.

	calculée/annulé du compte par l'opération distincte de balance	
DEAL_TYPE_CHARGE	Les taxes supplémentaires	HistoryDealGetInteger
DEAL_TYPE_COMMISSION	Les commissions supplémentaires	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_DAILY	Агентская комиссия, начисляемая в конце торгового дня	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Агентская комиссия, начисляемая в конце месяца	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_DAILY	La commission calculée à la fin du jour commercial	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_MONTHLY	La commission calculée à la fin du mois	HistoryDealGetInteger
DEAL_TYPE_CORRECTION	L'enregistrement corrigeant	HistoryDealGetInteger
DEAL_TYPE_CREDIT	La mise en compte du crédit	HistoryDealGetInteger
DEAL_TYPE_INTEREST	Начисления процентов на свободные средства	HistoryDealGetInteger
DEAL_TYPE_SELL	La vente	HistoryDealGetInteger
DEAL_TYPE_SELL_CANCELED	Le marché annulé de la vente. La situation est possible, quand le marché sur la vente auparavant fait est annulé. Dans ce cas, le type du marché auparavant fait (DEAL_TYPE_SELL) se change sur DEAL_TYPE_SELL_CANCELED, et son bénéfice/perte est remise à zéro. Le bénéfice/perte auparavant reçu est calculée/annulé du compte par l'opération distincte de balance	HistoryDealGetInteger
DEAL_VOLUME	Le volume du marché	HistoryDealGetDouble
DRAW_ARROW	Le dessin par les flèches	Styles du dessin
DRAW_BARS	L'affichage comme une séquence de barres	Styles du dessin
DRAW_CANDLES	L'affichage comme des bougies	Styles du dessin
DRAW_COLOR_ARROW	Le dessin par les flèches multicolores	Styles du dessin

© 2000-2015, MetaQuotes Software Corp.

DRAW_COLOR_BARS	Les barres multicolores	Styles du dessin
DRAW_COLOR_CANDLES	Les bougies multicolores	Styles du dessin
DRAW_COLOR_HISTOGRAM	L'histogramme multicolore de la ligne zéro	Styles du dessin
DRAW_COLOR_HISTOGRAM2	L'histogramme multicolore aux deux tampons d'indicateurs	Styles du dessin
DRAW_COLOR_LINE	La ligne multicolore	Styles du dessin
DRAW_COLOR_SECTION	Les segments multicolores	Styles du dessin
DRAW_COLOR_ZIGZAG	ZigZag multicolore	Styles du dessin
DRAW_FILLING	Le remplissage coloré entre les deux niveaux	Styles du dessin
DRAW_HISTOGRAM	L'histogramme de la ligne zéro	Styles du dessin
DRAW_HISTOGRAM2	L'histogramme des deux tampons d'indicateurs	Styles du dessin
DRAW_LINE	La ligne	Styles du dessin
DRAW_NONE	Ne dessine pas	Styles du dessin
DRAW_SECTION	Les segments	Styles du dessin
DRAW_ZIGZAG	Le style Zigzag admet les segments verticaux sur la barre	Styles du dessin
ELLIOTT_CYCLE	Une boucle (Cycle)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_GRAND_SUPERCYCLE	Un Supercycle principal (Grand Supercycle)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_INTERMEDIATE	Un chaînon intermédiaire (Intermediate)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINOR	Une boucle secondaire (Minor)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUETTE	Un seconde (Minuette)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUTE	Une minute (Minute)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_PRIMARY	Une boucle primaire (Primary)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUBMINUETTE	Un subseconde (Subminuette)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUPERCYCLE	Un Supercycle (Supercycle)	ObjectSetInteger, ObjectGetInteger

© 2000-2015, MetaQuotes Software Corp.

EMPTY_VALUE	La valeur vide dans le tampon d'indicateur	Les autres constantes
ERR_ACCOUNT_WRONG_PROPERTY	L'identificateur faux de la propriété du compte	GetLastError
ERR_ARRAY_BAD_SIZE	La grandeur de tableau demandé excède 2 gigoctets	GetLastError
ERR_ARRAY_RESIZE_ERROR	Il ne suffit pas la mémoire pour la redistribution du tableau ou la tentative du changement de la grandeur du tableau statique	GetLastError
ERR_BOOKS_CANNOT_ADD	Le profondeur de marché ne peut pas être ajouté	GetLastError
ERR_BOOKS_CANNOT_DELETE	Le profondeur de marché ne peut pas être supprimé	GetLastError
ERR_BOOKS_CANNOT_GET	Les données du profondeur de marché ne peuvent pas être reçues	GetLastError
ERR_BOOKS_CANNOT_SUBSCRIBE	L'erreur à la souscription sur la réception des nouvelles données du profondeur de marché	GetLastError
ERR_BUFFERS_NO_MEMORY	Il ne suffit pas la mémoire pour la distribution des tampons d'indicateur	GetLastError
ERR_BUFFERS_WRONG_INDEX	L'index faux du tampon d'indicateur	GetLastError
ERR_CANNOT_CLEAN_DIRECTORY	On n'a pas réussi à vider le répertoire (probablement, un ou quelques fichiers sont bloqués et l'opération de l'effacement n'a pas réussi)	GetLastError
ERR_CANNOT_DELETE_DIRECTORY	Le répertoire ne peut pas être supprimé	GetLastError
ERR_CANNOT_DELETE_FILE	L'erreur de l'effacement du fichier	GetLastError
ERR_CANNOT_OPEN_FILE	L'erreur de l'ouverture du fichier	GetLastError
ERR_CHAR_ARRAY_ONLY	Le tableau doit être du type char	GetLastError
ERR_CHART_CANNOT_CHANGE	L'erreur au changement de chart du symbole et la période	GetLastError

© 2000-2015, MetaQuotes Software Corp.

ERR_CHART_CANNOT_CREATE_TIMER	L'erreur à la création du minuteur	GetLastError
ERR_CHART_CANNOT_OPEN	L'erreur de l'ouverture de chart	GetLastError
ERR_CHART_INDICATOR_CANNOT_ADD	L'erreur à l'ajout de l'indicateur sur le graphique	GetLastError
ERR_CHART_INDICATOR_CANNOT_DEL	L'erreur à la suppression de l'indicateur du graphique	GetLastError
ERR_CHART_INDICATOR_NOT_FOUND	L'indicateur n'est pas trouvé sur le graphique indiqué	GetLastError
ERR_CHART_NAVIGATE_FAILED	L'erreur de la navigation dans le chart	GetLastError
ERR_CHART_NO_EXPERT	Le chart n'a pas d'expert, qui pourrait traiter l'événement	GetLastError
ERR_CHART_NO_REPLY	Le chart ne répond pas	GetLastError
ERR_CHART_NOT_FOUND	Le chart n'est pas trouvé	GetLastError
ERR_CHART_SCREENSHOT_FAILED	L'erreur à la création des screenshots	GetLastError
ERR_CHART_TEMPLATE_FAILED	L'erreur à l'application de la modèle	GetLastError
ERR_CHART_WINDOW_NOT_FOUND	La sous- fenêtre contenant l'indicateur indiqué, n'est pas trouvé	GetLastError
ERR_CHART_WRONG_ID	L'indificateur faux de chart	GetLastError
ERR_CHART_WRONG_PARAMETER	La valeur erronée du paramètre pour la fonction du travail avec le graphique	GetLastError
ERR_CHART_WRONG_PROPERTY	L'identificateur faux de la propriété du chart	GetLastError
ERR_CUSTOM_WRONG_PROPERTY	L'identificateur faux de la propriété de l'indicateur d'utilisateur	GetLastError
ERR_DIRECTORY_NOT_EXIST	Le répertoire n'existe pas	GetLastError
ERR_DOUBLE_ARRAY_ONLY	Le tableau doit être du type double	GetLastError
ERR_FILE_BINSTRINGSIZE	La dimension de chaîne doit être spécifiée, puisque le fichier est ouvert comme binaire	GetLastError
ERR_FILE_CACHEBUFFER_ERROR	Pas assez de mémoire pour le cache pour la lecture	GetLastError

© 2000-2015, MetaQuotes Software Corp.

ERR_FILE_CANNOT_REWRITE	Le fichier ne peut pas être réécrit	GetLastError
ERR_FILE_IS_DIRECTORY	Ce n'est pas un fichier, c'est un répertoire	GetLastError
ERR_FILE_ISNOT_DIRECTORY	C'est un fichier, pas un répertoire	GetLastError
ERR_FILE_NOT_EXIST	Le fichier n'existe pas	GetLastError
ERR_FILE_NOTBIN	Le fichier doit être ouvert comme binaire	GetLastError
ERR_FILE_NOTCSV	Le fichier doit être ouvert comme CSV	GetLastError
ERR_FILE_NOTTOREAD	Le fichier doit être ouvert pour la lecture	GetLastError
ERR_FILE_NOTTOWRITE	Le fichier doit être ouvert pour l'enregistrement	GetLastError
ERR_FILE_NOTTXT	Le fichier doit être ouvert comme le texte	GetLastError
ERR_FILE_NOTTXTORCSV	Le fichier doit être ouvert comme le texte ou CSV	GetLastError
ERR_FILE_READERROR	L'erreur de la lecture du fichier	GetLastError
ERR_FILE_WRITEERROR	On n'a pas réussi à enregistrer la ressource au fichier	GetLastError
ERR_FLOAT_ARRAY_ONLY	Le tableau doit être du type float	GetLastError
ERR_FTP_SEND_FAILED	On ne réussit pas à expédier le fichier selon ftp	GetLastError
ERR_FUNCTION_NOT_ALLOWED	La fonction systémique n'est pas autorisé pour l'appel	GetLastError
ERR_GLOBALVARIABLE_EXISTS	La variable globale du terminal de client avec un tel nom existe déjà	GetLastError
ERR_GLOBALVARIABLE_NOT_FOUND	La variable globale du terminal de client n'est pas trouvée	GetLastError
ERR_HISTORY_NOT_FOUND	L'histoire demandée n'est pas trouvée	GetLastError
ERR_HISTORY_WRONG_PROPERTY	L'identificateur faux de la propriété de l'histoire	GetLastError
ERR_INCOMPATIBLE_ARRAYS	Le copiage des tableaux incompatibles. Le tableau de ligne peut être copié	GetLastError

© 2000-2015, MetaQuotes Software Corp.

	seulement à celui de ligne, et le tableau numérique – à celui numérique	
ERR_INCOMPATIBLE_FILE	Pour les tableaux de ligne doit être un fichier de texte, pour les autres – le fichier binaire	GetLastError
ERR_INDICATOR_CANNOT_ADD	L'erreur à l' incrémentation de l'indicateur	GetLastError
ERR_INDICATOR_CANNOT_APPLY	L'indicateur ne peut pas être appliqué à un autre indicateur	GetLastError
ERR_INDICATOR_CANNOT_CREATE	L'indicateur ne peut pas être créé	GetLastError
ERR_INDICATOR_CUSTOM_NAME	Le premier paramètre dans le tableau doit être le nom de l'indicateur d'utilisateur	GetLastError
ERR_INDICATOR_DATA_NOT_FOUND	Les données demandées ne sont pas trouvées	GetLastError
ERR_INDICATOR_NO_MEMORY	Pas assez de mémoire pour ajouter l'indicateur	GetLastError
ERR_INDICATOR_PARAMETER_TYPE	Le type incorrect du paramètre dans le tableau à la création de l'indicateur	GetLastError
ERR_INDICATOR_PARAMETERS_MISSING	Les paramètres manquent à la création de l'indicateur	GetLastError
ERR_INDICATOR_UNKNOWN_SYMBOL	Le symbole inconnu	GetLastError
ERR_INDICATOR_WRONG_HANDLE	Ошибочный хэндл индикатора	GetLastError
ERR_INDICATOR_WRONG_INDEX	Le handle de l'indicateur erroné	GetLastError
ERR_INDICATOR_WRONG_PARAMETERS	La quantité incorrecte de paramètres à la création de l'indicateur	GetLastError
ERR_INT_ARRAY_ONLY	Le tableau doit être du type int	GetLastError
ERR_INTERNAL_ERROR	L'erreur inattendue intérieure	GetLastError
ERR_INVALID_ARRAY	Le tableau du type inconvenant, de la grandeur inconvenante ou u l'objet abîmé du tableau dynamique	GetLastError
ERR_INVALID_DATETIME	La valeur incorrecte de la date	GetLastError

© 2000-2015, MetaQuotes Software Corp.

	et/ou le temps	
ERR_INVALID_FILEHANDLE	Le fichier avec un tel handle a été fermé, ou ne s'ouvrait pas du tout	GetLastError
ERR_INVALID_PARAMETER	Le paramètre faux à l'appel intérieur de la fonction systémique	GetLastError
ERR_INVALID_POINTER	L'indicateur faux	GetLastError
ERR_INVALID_POINTER_TYPE	Le type de l'indicateur faux	GetLastError
ERR_LONG_ARRAY_ONLY	Le tableau doit être du type long	GetLastError
ERR_MAIL_SEND_FAILED	On ne réussit pas à expédier le message	GetLastError
ERR_MARKET_LASTTIME_UNKNOWN	Le temps du dernier tique est inconnu (il n'y avait pas de ticks)	GetLastError
ERR_MARKET_NOT_SELECTED	Le symbole n'est pas choisi dans MarketWatch	GetLastError
ERR_MARKET_SELECT_ERROR	L'erreur de l'ajout ou de la suppression du symbole dans MarketWatch	GetLastError
ERR_MARKET_UNKNOWN_SYMBOL	Le symbole inconnu	GetLastError
ERR_MARKET_WRONG_PROPERTY	L'identificateur faux de la propriété du symbole	GetLastError
ERR_MQL5_WRONG_PROPERTY	L'identificateur faux de la propriété du programme	GetLastError
ERR_NO_STRING_DATE	Il n'y a pas de date dans la chaîne	GetLastError
ERR_NOT_ENOUGH_MEMORY	Il ne suffit pas la mémoire pour l'exécution de la fonction systémique	GetLastError
ERR_NOTIFICATION_SEND_FAILED	On n'a pas réussi d'envoyer la notification	GetLastError
ERR_NOTIFICATION_TOO_FREQUENT	L'expédition trop fréquente des notifications	GetLastError
ERR_NOTIFICATION_WRONG_PARAMETER	Le paramètre non valide pour envoyer une notification - on a transmis une chaîne vide dans une fonction SendNotification() ou NULL	GetLastError

© 2000-2015, MetaQuotes Software Corp.

ERR_NOTIFICATION_WRONG_SETTINGS	Les réglages incorrectes des notifications dans le terminal (ID n'est pas indiqué ou la permission n'est pas exposée)	GetLastError
ERR_NOTINITIALIZED_STRING	La ligne n'est pas initialisée	GetLastError
ERR_NUMBER_ARRAYS_ONLY	Le tableau doit être numérique	GetLastError
ERR_OBJECT_ERROR	L'erreur au fonctionnement de l'objet graphique	GetLastError
ERR_OBJECT_GETDATE_FAILED	Il est impossible de recevoir la date correspondant à la valeur	GetLastError
ERR_OBJECT_GETVALUE_FAILED	Il est impossible de recevoir la valeur correspondante à la date	GetLastError
ERR_OBJECT_NOT_FOUND	L'objet graphique n'est pas trouvé	GetLastError
ERR_OBJECT_WRONG_PROPERTY	L'identificateur faux de la propriété de l'objet graphique	GetLastError
ERR_ONEDIM_ARRAYS_ONLY	Le tableau doit être unidimentionnel	GetLastError
ERR_OPENCL_BUFFER_CREATE	L'erreur de la création du tampon OpenCL	GetLastError
ERR_OPENCL_CONTEXT_CREATE	L'erreur à la création du contexte OpenCL	GetLastError
ERR_OPENCL_EXECUTE	L'erreur de l'exécution du programme OpenCL	GetLastError
ERR_OPENCL_INTERNAL	l'erreur intérieure à l'exécution d' OpenCL	GetLastError
ERR_OPENCL_INVALID_HANDLE	Le handleincorrect OpenCL	GetLastError
ERR_OPENCL_KERNEL_CREATE	L'erreur de la créationdu point-kernel de l'entrée OpenCL	GetLastError
ERR_OPENCL_NOT_SUPPORTED	Les fonctions OpenCL ne sont pas soutenues sur cet ordinateur	GetLastError
ERR_OPENCL_PROGRAM_CREATE	L'erreur pendant la compilation du programme OpenCL	GetLastError
ERR_OPENCL_QUEUE_CREATE	L'erreur de la création du tour de l'exécution dans OpenCL	GetLastError
ERR_OPENCL_SET_KERNEL_PARAMETER	L'erreur à l'installation des paramètres pour le kernel	GetLastError

© 2000-2015, MetaQuotes Software Corp.

	OpenCL (le point de l'entrée au programme OpenCL)	
ERR_OPENCL_TOO_LONG_KERNEL_NAME	Un trop long nom du point de l'entrée (le kernel OpenCL)	GetLastError
ERR_OPENCL_WRONG_BUFFER_OFFSET	Un décalage incorrect dans le tampon OpenCL	GetLastError
ERR_OPENCL_WRONG_BUFFER_SIZE	Une taille incorrecte du tampon OpenCL	GetLastError
ERR_PLAY_SOUND_FAILED	On ne réussit pas à reproduire le son	GetLastError
ERR_RESOURCE_NAME_DUPLICATED	a coïncidence des nomsdes ressources et dynamique etstatique	GetLastError
ERR_RESOURCE_NAME_IS_TOO_LONG	Le nom de la ressource dépasse 63 symboles	GetLastError
ERR_RESOURCE_NOT_FOUND	La ressource avec un tel nom à EX5 n'est pas trouvée	GetLastError
ERR_RESOURCE_UNSUPPOTED_TYPE	Le type non soutenu de la ressource ou la taille plus 16 Mb	GetLastError
ERR_SERIES_ARRAY	La série temporelle ne peut pas être utilisée	GetLastError
ERR_SHORT_ARRAY_ONLY	Le tableau doit être du type short	GetLastError
ERR_SMALL_ARRAY	Un trop petit tableau, la position de départ est à l'extérieur du tableau	GetLastError
ERR_SMALL_ASSERIES_ARRAY	Le tableau de la réception est déclaré comme AS_SERIES, et il est de la taille insuffisante	GetLastError
ERR_STRING_OUT_OF_MEMORY	Pas assez de mémoire pour la chaîne	GetLastError
ERR_STRING_RESIZE_ERROR	Il ne suffit pas la mémoire pour la redistribution de la ligne	GetLastError
ERR_STRING_SMALL_LEN	La longueur de chaîne est moins qu'attendue	GetLastError
ERR_STRING_TIME_ERROR	L'erreur de la conversion de la chaîne à la date	GetLastError
ERR_STRING_TOO_BIGNUMBER	Le nombre est trop grand, plus que ULONG_MAX	GetLastError

© 2000-2015, MetaQuotes Software Corp.

ERR_STRING_UNKNOWNTYPE	Le type de données inconnu à la conversion à la chaîne	GetLastError
ERR_STRING_ZEROADDED	0 est ajouté à la fin de la chaîne, l'opération est inutile	GetLastError
ERR_STRINGPOS_OUTOFRANGE	La position en dehors de la chaîne	GetLastError
ERR_STRUCT_WITHOBJECTS_ORCLASS	La structure contient les objets des lignes et/ou les tableaux dynamiques et/ou la structure avec tels objets et/ou les classes	GetLastError
ERR_SUCCESS	Операция выполнена успешно	GetLastError
ERR_TERMINAL_WRONG_PROPERTY	L'identificateur faux de la propriété du terminal	GetLastError
ERR_TOO_LONG_FILENAME	Un trop long nom du fichier	GetLastError
ERR_TOO_MANY_FILES	Ne peut pas être ouvert simultanément plus de 64 fichiers	GetLastError
ERR_TOO_MANY_FORMATTERS	Plus de spécificateurs de format que des paramètres	GetLastError
ERR_TOO_MANY_PARAMETERS	Plus de paramètres que des spécificateurs de format	GetLastError
ERR_TRADE_DEAL_NOT_FOUND	Le marché n'est pas trouvé	GetLastError
ERR_TRADE_DISABLED	Le commerce pour l'expert est interdit	GetLastError
ERR_TRADE_ORDER_NOT_FOUND	L'ordre n'est pas trouvée	GetLastError
ERR_TRADE_POSITION_NOT_FOUND	La position n'est pas trouvée	GetLastError
ERR_TRADE_SEND_FAILED	On n'a pas réussi à expédier la requête commerciale	GetLastError
ERR_TRADE_WRONG_PROPERTY	L'identificateur faux de la propriété du commerce	GetLastError
ERR_USER_ERROR_FIRST	Par ce code commencent les erreurs, spécifiées par l'utilisateur	GetLastError
ERR_WEBREQUEST_CONNECT_FAILED	Не удалось подключиться к указанному URL	GetLastError
ERR_WEBREQUEST_INVALID_A	URL не прошел проверку	GetLastError

© 2000-2015, MetaQuotes Software Corp.

DDRESS		
ERR_WEBREQUEST_REQUEST_FAILED	Ошибка в результате выполнения HTTP запроса	GetLastError
ERR_WEBREQUEST_TIMEOUT	Превышен таймаут получения данных	GetLastError
ERR_WRONG_DIRECTORYNAME	Le nom faux du répertoire	GetLastError
ERR_WRONG_FILEHANDLE	Le handle faux du fichier	GetLastError
ERR_WRONG_FILENAME	Le nom inadmissible du fichier	GetLastError
ERR_WRONG_FORMATSTRING	La chaîne fausse de format	GetLastError
ERR_WRONG_INTERNAL_PARAMETER	Le paramètre faux à l'appel intérieur de la fonction du terminal de client	GetLastError
ERR_WRONG_STRING_DATE	Une date fausse dans la chaîne	GetLastError
ERR_WRONG_STRING_OBJECT	L'objet abîmé de la chaîne	GetLastError
ERR_WRONG_STRING_PARAMETER	Le paramètre abîmé du type string	GetLastError
ERR_WRONG_STRING_TIME	Un temps faux dans la chaîne	GetLastError
ERR_ZEROSIZE_ARRAY	Le tableau de la longueur nulle	GetLastError
FILE_ACCESS_DATE	La date du dernier accès au fichier	FileGetInteger
FILE_ANSI	Les chaînes du type ANSI (les symboles d'un byte). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())	FileOpen
FILE_BIN	Le mode binaire de la lecture-enregistrement (sans conversion de la chaîne et à la chaîne). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())	FileOpen
FILE_COMMON	La disposition du fichier dans le dossier commun de tous les terminaux de client \Terminal \Common\Files. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()), au copiage des fichiers (FileCopy(), FileMove()) et à la vérification de l'existence des	FileOpen, FileCopy, FileMove, FileIsExist

© 2000-2015, MetaQuotes Software Corp.

	fichiers (FileIsExist())	
FILE_CREATE_DATE	La date de la création	FileGetInteger
FILE_CSV	Le fichier du type csv (Tous les éléments inscrits seront transcrits vers les chaînes du type correspondant, unicode ou ansi, et se divisent par le séparateurU). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())	FileOpen
FILE_END	La réception du critère de la fin du fichier	FileGetInteger
FILE_EXISTS	La vérification de l'existence	FileGetInteger
FILE_IS_ANSI	Le fichier est ouvert comme ANSI (à voir FILE_ANSI)	FileGetInteger
FILE_IS_BINARY	Le fichier est ouvert comme binaire (à voir FILE_BIN)	FileGetInteger
FILE_IS_COMMON	Le fichier est ouvert dans un dossier commun de tous les terminaux de client (à voir FILE_COMMON)	FileGetInteger
FILE_IS_CSV	Le fichier est ouvert comme CSV (à voir FILE_CSV)	FileGetInteger
FILE_IS_READABLE	Le fichier est ouvert avec la possibilité de la lecture (à voir FILE_READ)	FileGetInteger
FILE_IS_TEXT	Le fichier est ouvert comme celui de texte (à voir FILE_TXT)	FileGetInteger
FILE_IS_WRITABLE	Le fichier est ouvert avec la possibilité de l'enregistrement (à voir FILE_WRITE)	FileGetInteger
FILE_LINE_END	La réception du critère de la fin de la ligne	FileGetInteger
FILE_MODIFY_DATE	La date du dernier changement	FileGetInteger
FILE_POSITION	La position du pointeur au fichier	FileGetInteger
FILE_READ	Le fichier s'ouvre pour la lecture. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()). A l'ouverture du	FileOpen

© 2000-2015, MetaQuotes Software Corp.

	fichier on doit indiquer absolument le drapeau FILE_WRITE et/ou le drapeux FILE_READ.	
FILE_REWRITE	La possibilité de réécrire le fichier par les fonctions FileCopy() et FileMove(). Le fichier doit exister ou s'ouvrir pour l'enregistrement. Dans le cas contraire le fichier ne sera pas ouvert	FileCopy, FileMove
FILE_SHARE_READ	L'accès commun selon la lecture de plusieurs programmes. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()), mais ne remplace pas à l'ouverture du fichier la nécessité d'indiquer FILE_WRITE et/ou le drapeau FILE_READ	FileOpen
FILE_SHARE_WRITE	L'accès commun selon l'enregistrement de plusieurs programmes. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()), mais ne remplace pas à l'ouverture du fichier la nécessité d'indiquer FILE_WRITE et/ou le drapeau FILE_READ	FileOpen
FILE_SIZE	La taille du fichier en bytes	FileGetInteger
FILE_TXT	Un simple fichier de texte (le même csv, cependant le séparateur n'est pas pris en considération). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())	FileOpen
FILE_UNICODE	Les chaînes du type UNICODE (les symboles de deux bytes). Le drapeau est utilisé à l'ouverture des fichiers (FileOpen())	FileOpen
FILE_WRITE	Le fichier s'ouvre pour l'enregistrement. Le drapeau est utilisé à l'ouverture des fichiers (FileOpen()). A l'ouverture du fichier on doit	FileOpen

© 2000-2015, MetaQuotes Software Corp.

	indiquer absolument le drapeau FILE_WRITE et/ou le drapeux FILE_READ.	
FLT_DIG	Le nombre de signes signifiants décimaux	Constantes des types de données numériques
FLT_EPSILON	La valeur minimale, qui satisfait à la condition : 1.0 +FLT_EPSILON != 1.0	Constantes des types de données numériques
FLT_MANT_DIG	Le nombre de bits dans la mantisse	Constantes des types de données numériques
FLT_MAX	La valeur maximale, qui peut être présentée par le type float	Constantes des types de données numériques
FLT_MAX_10_EXP	La valeur maximale décimale du degré d'exposant	Constantes des types de données numériques
FLT_MAX_EXP	La valeur maximale binaire du degré d'exposant	Constantes des types de données numériques
FLT_MIN	La valeur minimale positive, qui peut être présentée par le type float	Constantes des types de données numériques
FLT_MIN_10_EXP	La valeur minimale décimale du degré d'exposant	Constantes des types de données numériques
FLT_MIN_EXP	La valeur minimale binaire du degré d'exposant	Constantes des types de données numériques
FRIDAY	Vendredi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
GANN_DOWN_TREND	La ligne correspond à la tendance descendante	ObjectSetInteger, ObjectGetInteger
GANN_UP_TREND	La ligne correspond à la tendance montante	ObjectSetInteger, ObjectGetInteger
GATORJAW_LINE	La ligne de mâchoire	Lignes des indicateurs
GATORLIPS_LINE	La ligne de lèvres	Lignes des indicateurs
GATORTEETH_LINE	La ligne de dents	Lignes des indicateurs
IDABORT	On a choisi le bouton Interrompre (Abort)	MessageBox
IDCANCEL	On a choisi le bouton Suppression (Cancel)	MessageBox
IDCONTINUE	On a choisi le bouton Continuer (Continue)	MessageBox

© 2000-2015, MetaQuotes Software Corp.

IDIGNORE	On a choisi le bouton Ignorer (Ignore)	MessageBox
IDNO	On a choisi le bouton No (No)	MessageBox
IDOK	On a choisi le bouton OK	MessageBox
IDRETRY	On a choisi le bouton Répétition (Retry)	MessageBox
IDTRYAGAIN	On a choisi le bouton Répéter (Try Again)	MessageBox
IDYES	On a choisi le bouton Oui (Yes)	MessageBox
IND_AC	Accelerator Oscillator	IndicatorCreate, IndicatorParameters
IND_AD	Accumulation/Distribution	IndicatorCreate, IndicatorParameters
IND_ADX	Average Directional Index	IndicatorCreate, IndicatorParameters
IND_ADXW	ADX by Welles Wilder	IndicatorCreate, IndicatorParameters
IND_ALLIGATOR	Alligator	IndicatorCreate, IndicatorParameters
IND_AMA	Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_AO	Awesome Oscillator	IndicatorCreate, IndicatorParameters
IND_ATR	Average True Range	IndicatorCreate, IndicatorParameters
IND_BANDS	Bollinger Bands®	IndicatorCreate, IndicatorParameters
IND_BEARS	Bears Power	IndicatorCreate, IndicatorParameters
IND_BULLS	Bulls Power	IndicatorCreate, IndicatorParameters
IND_BWMFI	Market Facilitation Index	IndicatorCreate, IndicatorParameters
IND_CCI	Commodity Channel Index	IndicatorCreate, IndicatorParameters
IND_CHAIKIN	Chaikin Oscillator	IndicatorCreate, IndicatorParameters
IND_CUSTOM	Custom indicator	IndicatorCreate, IndicatorParameters

© 2000-2015, MetaQuotes Software Corp.

IND_DEMA	Double Exponential Moving Average	IndicatorCreate, IndicatorParameters
IND_DEMARKER	DeMarker	IndicatorCreate, IndicatorParameters
IND_ENVELOPES	Envelopes	IndicatorCreate, IndicatorParameters
IND_FORCE	Force Index	IndicatorCreate, IndicatorParameters
IND_FRACTALS	Fractals	IndicatorCreate, IndicatorParameters
IND_FRAMA	Fractal Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_GATOR	Gator Oscillator	IndicatorCreate, IndicatorParameters
IND_ICHIMOKU	Ichimoku Kinko Hyo	IndicatorCreate, IndicatorParameters
IND_MA	Moving Average	IndicatorCreate, IndicatorParameters
IND_MACD	MACD	IndicatorCreate, IndicatorParameters
IND_MFI	Money Flow Index	IndicatorCreate, IndicatorParameters
IND_MOMENTUM	Momentum	IndicatorCreate, IndicatorParameters
IND_OBV	On Balance Volume	IndicatorCreate, IndicatorParameters
IND_OSMA	OsMA	IndicatorCreate, IndicatorParameters
IND_RSI	Relative Strength Index	IndicatorCreate, IndicatorParameters
IND_RVI	Relative Vigor Index	IndicatorCreate, IndicatorParameters
IND_SAR	Parabolic SAR	IndicatorCreate, IndicatorParameters
IND_STDDEV	Standard Deviation	IndicatorCreate, IndicatorParameters
IND_STOCHASTIC	Stochastic Oscillator	IndicatorCreate, IndicatorParameters
IND_TEMA	Triple Exponential Moving Average	IndicatorCreate, IndicatorParameters

© 2000-2015, MetaQuotes Software Corp.

IND_TRIX	Triple Exponential Moving Averages Oscillator	IndicatorCreate, IndicatorParameters
IND_VIDYA	Variable Index Dynamic Average	IndicatorCreate, IndicatorParameters
IND_VOLUMES	Volumes	IndicatorCreate, IndicatorParameters
IND_WPR	Williams' Percent Range	IndicatorCreate, IndicatorParameters
INDICATOR_CALCULATIONS	Les tampons auxiliaires pour les calculs intermédiaires	SetIndexBuffer
INDICATOR_COLOR_INDEX	Les couleurs du dessin	SetIndexBuffer
INDICATOR_DATA	Les données pour le dessin	SetIndexBuffer
INDICATOR_DIGITS	L'exactitude de l'affichage des valeurs de l'indicateur	IndicatorSetInteger
INDICATOR_HEIGHT	La hauteur fixée de la fenêtre personnelle de l'indicateur (la commande du préprocesseur#property indicator_height)	IndicatorSetInteger
INDICATOR_LEVELCOLOR	La couleur de la ligne du niveau	IndicatorSetInteger
INDICATOR_LEVELS	Le nombre de niveaux dans la fenêtre de l'indicateur	IndicatorSetInteger
INDICATOR_LEVELSTYLE	Le style de la ligne du niveau	IndicatorSetInteger
INDICATOR_LEVELTEXT	La description du niveau	IndicatorSetString
INDICATOR_LEVELVALUE	La valeur du niveau	IndicatorSetDouble
INDICATOR_LEVELWIDTH	L'épaisseur de la ligne du niveau	IndicatorSetInteger
INDICATOR_MAXIMUM	Le maximum de la fenêtre de l'indicateur	IndicatorSetDouble
INDICATOR_MINIMUM	Le minimum de la fenêtre de l'indicateur	IndicatorSetDouble
INDICATOR_SHORTNAME	Le nom court de l'indicateur	IndicatorSetString
INT_MAX	La valeur maximale, qui peut être présentée par le type int	Constantes des types de données numériques
INT_MIN	La valeur minimale, qui peut être présentée par le type int	Constantes des types de données numériques
INVALID_HANDLE	Le handle incorrect	Les autres constantes

© 2000-2015, MetaQuotes Software Corp.

IS_DEBUG_MODE	Le critère du travail du programme mq5 en mode du débogage	Les autres constantes
IS_PROFILE_MODE	Le critère du travail du programme mq5 en mode du profilage	Les autres constantes
KIJUNSEN_LINE	La ligne Kijun-sen	Lignes des indicateurs
LICENSE_DEMO	La version démo du produit payant du Market Fonctionne seulement dans le testeur des stratégies	MQLInfoInteger
LICENSE_FREE	La version gratuite non limitée	MQLInfoInteger
LICENSE_FULL	Купленная лицензионная версия допускает не менее 5 активаций. Продавец может увеличить разрешенное число активаций	MQLInfoInteger
LICENSE_TIME	Версия с ограниченной по времени лицензией	MQLInfoInteger
LONG_MAX	La valeur maximale, qui peut être présentée par le type long	Constantes des types de données numériques
LONG_MIN	La valeur minimale, qui peut être présentée par le type long	Constantes des types de données numériques
LOWER_BAND	Une limite inférieure	Lignes des indicateurs
LOWER_HISTOGRAM	L'histogramme inférieur	Lignes des indicateurs
LOWER_LINE	Une ligne inférieure	Lignes des indicateurs
M_1_PI	1/pi	Constantes mathématiques
M_2_PI	2/pi	Constantes mathématiques
M_2_SQRTPI	2/sqrt(pi)	Constantes mathématiques
M_E	e	Constantes mathématiques
M_LN10	ln(10)	Constantes mathématiques
M_LN2	ln(2)	Constantes mathématiques
M_LOG10E	log10(e)	Constantes mathématiques
M_LOG2E	log2(e)	Constantes mathématiques
M_PI	pi	Constantes mathématiques
M_PI_2	pi/2	Constantes mathématiques
M_PI_4	pi/4	Constantes mathématiques

© 2000-2015, MetaQuotes Software Corp.

M_SQRT1_2	1/sqrt(2)	Constantes mathématiques
M_SQRT2	sqrt(2)	Constantes mathématiques
MAIN_LINE	La ligne principale	Lignes des indicateurs
MB_ABORTRETRYIGNORE	La fenêtre du message contient trois boutons: Abort, Retry et Ignore	MessageBox
MB_CANCELTRYCONTINUE	La fenêtre du message contient trois boutons: Cancel, Try Again, Continue	MessageBox
MB_DEFBUTTON1	Le premier bouton MB_DEFBUTTON1 - le bouton est choisi par défaut, si MB_DEFBUTTON2, MB_DEFBUTTON3, ou MB_DEFBUTTON4 ne sont pas définis	MessageBox
MB_DEFBUTTON2	Le deuxième bouton - le bouton par défaut	MessageBox
MB_DEFBUTTON3	Le troisième bouton - le bouton par défaut	MessageBox
MB_DEFBUTTON4	Le quatrième bouton - le bouton par défaut	MessageBox
MB_ICONEXCLAMATION, MB_ICONWARNING	L'icône de signe de question	MessageBox
MB_ICONINFORMATION, MB_ICONASTERISK	L'icône, comprenant le signe de ligne i dans le cercle	MessageBox
MB_ICONQUESTION	L'icône de signe de question	MessageBox
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	L'icône de signe STOP	MessageBox
MB_OK	La fenêtre du message contient un bouton: OK. Par défaut	MessageBox
MB_OKCANCEL	La fenêtre du message contient deux boutons: OK et Cancel	MessageBox
MB_RETRYCANCEL	La fenêtre du message contient deux boutons: Retry et Cancel	MessageBox
MB_YESNO	La fenêtre du message contient deux boutons: Yes et	MessageBox

© 2000-2015, MetaQuotes Software Corp.

	No	
MB_YESNOCANCEL	La fenêtre du message contient trois boutons: Yes, No et Cancel	MessageBox
MINUSDI_LINE	La ligne -DI	Lignes des indicateurs
MODE_EMA	La prise de moyenne exponentielle	Méthodes des glissantes
MODE_LWMA	La prise de moyenne de ligne-pesée	Méthodes des glissantes
MODE_SMA	La prise de moyenne simple	Méthodes des glissantes
MODE_SMMA	La prise de moyenne lissée	Méthodes des glissantes
MONDAY	Lundi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
MQL_DEBUG	L'indice du travail du programme exécuté au mode du débogage	MQLInfoInteger
MQL_DLLS_ALLOWED	La permission de l'utilisation DLL pour ce programme exécuté	MQLInfoInteger
MQL_FRAME_MODE	Le critère du travail de l'expert lancé sur le graphique enmode du rassemblement des trames des résultats de l'optimisation	MQLInfoInteger
MQL_LICENSE_TYPE	Le type de la licence du module EX5. La licence se rapporte notamment à ce module EX5, d'où on fait la demande à l'aide de MQLInfoInteger(MQL_LICENSE_TYPE).	MQLInfoInteger
MQL_MEMORY_LIMIT	La quantité maximale possible de mémoire dynamique pour le programme MQL5 en Mb	MQLInfoInteger
MQL_MEMORY_USED	La taille de la mémoire utilisée par le programme MQL5 en Mb	MQLInfoInteger
MQL_OPTIMIZATION	L'indice du travail du programme exécuté en train de l'optimisation	MQLInfoInteger
MQL_PROFILER	Le critère du travail du programme lancé en mode du profilage du code	MQLInfoInteger

© 2000-2015, MetaQuotes Software Corp.

MQL_PROGRAM_NAME	Le nom du programme mql5 exécuté	MQLInfoString
MQL_PROGRAM_PATH	Le chemin pour ce programme exécuté	MQLInfoString
MQL_PROGRAM_TYPE	Le type de programme mql5	MQLInfoInteger
MQL_SIGNALS_ALLOWED	Разрешение на работу с сигналами данной запущенной программы	MQLInfoInteger
MQL_TESTER	L'indicatif du travail du programme exécuté au testeur	MQLInfoInteger
MQL_TRADE_ALLOWED	La permission du commerce pour ce programme exécuté	MQLInfoInteger
MQL_VISUAL_MODE	L'indice du travail du programme exécuté dans le mode visuel de test	MQLInfoInteger
NULL	Le zéro de n'importe quel type	Les autres constantes
OBJ_ALL_PERIODS	L'objet se dessine sur tous les temps trames	ObjectSetInteger, ObjectGetInteger
OBJ_ARROW	L'objet "La flèche"	Types des objets
OBJ_ARROW_BUY	Le signe "Buy"	Types des objets
OBJ_ARROW_CHECK	Le signe "Vérification"	Types des objets
OBJ_ARROW_DOWN	Le signe "la flèche en bas"	Types des objets
OBJ_ARROW_LEFT_PRICE	La balise gauche de prix	Types des objets
OBJ_ARROW_RIGHT_PRICE	La balise droite de prix	Types des objets
OBJ_ARROW_SELL	Le signe "Sell"	Types des objets
OBJ_ARROW_STOP	Le signe "Stop"	Types des objets
OBJ_ARROW_THUMB_DOWN	Le signe "mal" (le pouce en bas)	Types des objets
OBJ_ARROW_THUMB_UP	Le signe "bien" (le pouce en haut)	Types des objets
OBJ_ARROW_UP	Le signe "la flèche en haut"	Types des objets
OBJ_ARROWED_LINE	L'objet "Line avec une flèche"	Types des objets
OBJ_BITMAP	L'objet "Le dessin"	Types des objets
OBJ_BITMAP_LABEL	L'objet "La balise graphique"	Types des objets
OBJ_BUTTON	L'objet "Le bouton"	Types des objets
OBJ_CHANNEL	Le canal équidistant	Types des objets

© 2000-2015, MetaQuotes Software Corp.

OBJ_CHART	L'objet "Le graphique"	Types des objets
OBJ_CYCLES	Les lignes cycliques	Types des objets
OBJ_EDIT	L'objet "Le champ de l'entrée"	Types des objets
OBJ_ELLIOTWAVE3	Vague d'Elliott corrective	Types des objets
OBJ_ELLIOTWAVE5	Vague d'Elliott pulsatoire	Types des objets
OBJ_ELLIPSE	L'ellipse	Types des objets
OBJ_EVENT	L'objet "Événement", correspondant à l'événement dans le calendrier économique	Types des objets
OBJ_EXPANSION	L'expansion de Fibonacci	Types des objets
OBJ_FIBO	Les niveaux de Fibonacci	Types des objets
OBJ_FIBOARC	Les arcs de Fibonacci	Types des objets
OBJ_FIBOCHANNEL	Le canale de Fibonacci	Types des objets
OBJ_FIBOFAN	Le fanion de Fibonacci	Types des objets
OBJ_FIBOTIMES	Les zones temporaires de Fibonacci	Types des objets
OBJ_GANNFAN	Le fanion de Gann	Types des objets
OBJ_GANNGRID	La grille de Gann	Types des objets
OBJ_GANNLINE	La ligne de Gann	Types des objets
OBJ_HLINE	Ligne horizontale	Types des objets
OBJ_LABEL	L'objet "La balise de texte"	Types des objets
OBJ_NO_PERIODS	Объект не показывается ни на одном таймфрейме	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_D1	L'objet se dessine sur les graphiques de jour	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H1	L'objet se dessine sur les graphiques de 1 heure	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H12	L'objet se dessine sur les graphiques de 12 heures	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H2	L'objet se dessine sur les graphiques de 2 heures	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H3	L'objet se dessine sur les graphiques de 3 heures	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H4	L'objet se dessine sur les graphiques de 4 heures	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H6	L'objet se dessine sur les	ObjectSetInteger,

© 2000-2015, MetaQuotes Software Corp.

	graphiques de 6 heures	ObjectGetInteger
OBJ_PERIOD_H8	L'objet se dessine sur les graphiques de 8 heures	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M1	L'objet se dessine sur les graphiques de 1 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M10	L'objet se dessine sur les graphiques de 10 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M12	L'objet se dessine sur les graphiques de 12 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M15	L'objet se dessine sur les graphiques de 15 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M2	L'objet se dessine sur les graphiques de 2 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M20	L'objet se dessine sur les graphiques de 20 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M3	L'objet se dessine sur les graphiques de 3 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M30	L'objet se dessine sur les graphiques de 30 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M4	L'objet se dessine sur les graphiques de 4 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M5	L'objet se dessine sur les graphiques de 5 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M6	L'objet se dessine sur les graphiques de 6 minutes	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_MN1	L'objet se dessine sur les graphiques des mois	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_W1	L'objet se dessine sur les graphiques d'une semaine	ObjectSetInteger, ObjectGetInteger
OBJ_PITCHFORK	La fourche d'Andrews	Types des objets
OBJ_RECTANGLE	Le rectangle	Types des objets
OBJ_RECTANGLE_LABEL	L'objet "Marque rectangulaire" pour la création et la présentation de l'interface d'utilisateur graphique.	Types des objets
OBJ_REGRESSION	Le canale de la régression linéaire	Types des objets
OBJ_STDDEVCHANNEL	Le canal de l' écart type	Types des objets
OBJ_TEXT	L'objet "Le texte"	Types des objets

© 2000-2015, MetaQuotes Software Corp.

OBJ_TREND	La ligne de Tendance	Types des objets
OBJ_TRENDBYANGLE	La ligne de tendance par l'angle	Types des objets
OBJ_TRIANGLE	Le triangle	Types des objets
OBJ_VLINE	La ligne verticale	Types des objets
OBJPROP_ALIGN	L'alignement du texte à l'horizontale dans l'objet "Champ de l'entrée" (OBJ_EDIT)	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANCHOR	La position du point du rattachement de l'objet graphique	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANGLE	L'angle. Pour les objets créés du programme avec l'angle qui ne pas encore spécifié, la valeur est égale à EMPTY_VALUE	ObjectSetDouble, ObjectGetDouble
OBJPROP_ARROWCODE	Le code de la flèche pour l'objet "La flèche"	ObjectSetInteger, ObjectGetInteger
OBJPROP_BACK	L'objet à l'arrière-plan	ObjectSetInteger, ObjectGetInteger
OBJPROP_BGCOLOR	La couleur du fond pour OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL	ObjectSetInteger, ObjectGetInteger
OBJPROP_BMPFILE	Le nom du fichier BMP pour l'objet "Une balise graphique". Voir aussi Ressources	ObjectSetString, ObjectGetString
OBJPROP_BORDER_COLOR	La couleur du cadre pour l'objet OBJ_EDIT et OBJ_BUTTON	ObjectSetInteger, ObjectGetInteger
OBJPROP_BORDER_TYPE	Le type du cadre pour l'objet "Le cadre rectangulaire"	ObjectSetInteger, ObjectGetInteger
OBJPROP_CHART_ID	L'identificateur de l'objet "Graphique" (OBJ_CHART). Permet de travailler avec les propriétés de cet objet comme avec le graphique ordinaire à l'aide des fonctions du paragraphe Les opérations avec les graphiques, mais il y a certains exceptions.	ObjectSetInteger, ObjectGetInteger
OBJPROP_CHART_SCALE	L'échelle pour l'objet "le	ObjectSetInteger,

© 2000-2015, MetaQuotes Software Corp.

	graphique"	ObjectGetInteger
OBJPROP_COLOR	La couleur	ObjectSetInteger, ObjectGetInteger
OBJPROP_CORNER	L'angle du graphique pour le rattachement de l'objet graphique	ObjectSetInteger, ObjectGetInteger
OBJPROP_CREATETIME	Le temps de la création de l'objet	ObjectSetInteger, ObjectGetInteger
OBJPROP_DATE_SCALE	Affichage de l'échelle du temps pour l'objet "Graphique"	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEGREE	Le niveau du marquage d'onde d'Elliott	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEVIATION	La déviation pour le canal de la déviation standard	ObjectSetDouble, ObjectGetDouble
OBJPROP_DIRECTION	La tendance de l'objet de Gann	ObjectSetInteger, ObjectGetInteger
OBJPROP_DRAWLINES	L'affichage des lignes pour le marquage d'onde d'Elliott	ObjectSetInteger, ObjectGetInteger
OBJPROP_ELLIPSE	L'affichage de l'ellipse complète pour l'objet "l'Arc de Fibonacci" (OBJ_FIBOARC)	ObjectSetInteger, ObjectGetInteger
OBJPROP_FILL	Le remplissage d'un objet avec la couleur (pour OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)	ObjectSetInteger, ObjectGetInteger
OBJPROP_FONT	La fonte	ObjectSetString, ObjectGetString
OBJPROP_FONTSIZE	La dimension de la fonte	ObjectSetInteger, ObjectGetInteger
OBJPROP_HIDDEN	L'interdiction de montrer le nom d'un objet graphique dans la liste des objets du menu du terminal "Graphiques" - "Objets" - "Liste des objets". La valeur true permet de cacher l'objet inutile pour l'utilisateur de la liste. Par défaut true est défini pour les objets, qui affichent les événements du calendrier,	ObjectSetInteger, ObjectGetInteger

© 2000-2015, MetaQuotes Software Corp.

	l'histoire du commerce, ainsi que pour les objets créés du programme. Pour voir tels objets graphiques et recevoir l'accès à leurs propriétés, il faut appuyer sur le bouton "Tout" dans la fenêtre "Liste des objets".	
OBJPROP_LEVELCOLOR	La couleur de la ligne-niveau	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELS	Le nombre de niveaux	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELSTYLE	Le style de la ligne-niveau	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELTEXT	La description du niveau	ObjectSetString, ObjectGetString
OBJPROP_LEVELVALUE	La valeur du niveau	ObjectSetDouble, ObjectGetDouble
OBJPROP_LEVELWIDTH	L'épaisseur de la ligne-niveau	ObjectSetInteger, ObjectGetInteger
OBJPROP_NAME	Le nom de l'objet	ObjectSetString, ObjectGetString
OBJPROP_PERIOD	La période pour l'objet "le graphique"	ObjectSetInteger, ObjectGetInteger
OBJPROP_PRICE	La coordonnée du prix	ObjectSetDouble, ObjectGetDouble
OBJPROP_PRICE_SCALE	Affichage de l'échelle de prix pour l'objet "Graphique"	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY	Une ligne verticale passe par toutes les fenêtres d'un graphique	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_LEFT	Le rayon va à gauche	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_RIGHT	Le rayon va à droite	ObjectSetInteger, ObjectGetInteger
OBJPROP_READONLY	La possibilité de l'édition du texte dans l'objet Edit	ObjectSetInteger, ObjectGetInteger
OBJPROP_SCALE	L'échelle (la propriété des objets de Gann et l'objet "les arcs de Fibonacci")	ObjectSetDouble, ObjectGetDouble
OBJPROP_SELECTABLE	L'accessibilité de l'objet	ObjectSetInteger,

© 2000-2015, MetaQuotes Software Corp.

		ObjectGetInteger
OBJPROP_SELECTED	L'objet est sélectionné	ObjectSetInteger, ObjectGetInteger
OBJPROP_STATE	L'état du bouton (est appuyé/ est pressé)	ObjectSetInteger, ObjectGetInteger
OBJPROP_STYLE	Le style	ObjectSetInteger, ObjectGetInteger
OBJPROP_SYMBOL	Le caractère pour l'objet "le graphique"	ObjectSetString, ObjectGetString
OBJPROP_TEXT	La description de l'objet (le texte qui se trouve dans l'objet)	ObjectSetString, ObjectGetString
OBJPROP_TIME	La coordonnée du temps	ObjectSetInteger, ObjectGetInteger
OBJPROP_TIMEFRAMES	La visibilité de l'objet sur les temps trames	ObjectSetInteger, ObjectGetInteger
OBJPROP_TOOLTIP	Le texte de l'infobulle émergeante. Si la propriété n'est pas spécifiée, se montre l'infobulle automatiquement formée par le terminal. On peut désactiver l'infobulle, en définissant sa valeur "\ n" (saut de la ligne)	ObjectSetString, ObjectGetString
OBJPROP_TYPE	Le type de l'objet	ObjectSetInteger, ObjectGetInteger
OBJPROP_WIDTH	L'épaisseur de la ligne	ObjectSetInteger, ObjectGetInteger
OBJPROP_XDISTANCE	La distance en pixels selon l'axe X de l'angle du rattachement (см. примечание)	ObjectSetInteger, ObjectGetInteger
OBJPROP_XOFFSET	La coordonnée X de l'angle supérieur gauche du domaine rectangulaire de la visibilité dans les objets graphiques "Marque graphique" et "Dessin" (OBJ_BITMAP_LABEL et OBJ_BITMAP). La valeur est spécifiée en pixels par rapport à l'angle supérieur gauche de l'image initiale.	ObjectSetInteger, ObjectGetInteger
OBJPROP_XSIZE	La largeur de l'objet selon l'axe	ObjectSetInteger,

© 2000-2015, MetaQuotes Software Corp.

	X en pixels. Est spécifiée pour les objets OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	ObjectGetInteger
OBJPROP_YDISTANCE	La distance en pixels selon l'axe Y de l'angle du rattachement (см. примечание)	ObjectSetInteger, ObjectGetInteger
OBJPROP_YOFFSET	La coordonnée Y de l'angle supérieur gauche du domaine rectangulaire de la visibilité dans les objets graphiques "Marque graphique" et "Dessin" (OBJ_BITMAP_LABEL et OBJ_BITMAP). La valeur est spécifiée en pixels par rapport à l'angle supérieur gauche de l'image initiale.	ObjectSetInteger, ObjectGetInteger
OBJPROP_YSIZE	La hauteur de l'objet selon l'axe Y en pixels. Est spécifiée pour les objets OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	ObjectSetInteger, ObjectGetInteger
OBJPROP_ZORDER	La priorité de l'objet graphique sur la réception de l'événement de clic de la souris sur le graphique (CHARTEVENT_CLICK). Par défaut la valeur est égal au zéro à la création, mais la priorité peut être augmentée en cas de nécessité. En appliquant des objets un sur un autre, seulement un objet avec la plus haute priorité recevra l'événement CHARTEVENT_CLICK.	ObjectSetInteger, ObjectGetInteger
ORDER_COMMENT	Le commentaire	OrderGetString, HistoryOrderGetString
ORDER_FILLING_FOK	Cette politique de l'exécution signifie que l'ordre peut être	OrderGetInteger, HistoryOrderGetInteger

© 2000-2015, MetaQuotes Software Corp.

	exceptionnellement exécuté dans le volume indiqué. Si au marché au moment donné il n'y a pas de volume suffisant de l'instrument financier, l'ordre ne sera pas exécuté. Le volume nécessaire peut être fait de quelques propositions accessibles au moment donné sur le marché.	
ORDER_FILLING_IOC	Signifie l'accord de passer le marché par le volume au maximum accessible sur le marché indiqué à l'ordre. En cas d'impossibilité de l'exécution complète l'ordre sera exécuté sur le volume accessible, et le volume non exécuté de l'ordre sera annulé.	OrderGetInteger, HistoryOrderGetInteger
ORDER_FILLING_RETURN	Ce mode est utilisé pour les ordres de marché(ORDER_TYPE_BUY et ORDER_TYPE_SELL), limites et de stop de limite (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT) et seulement en modes "Exécution selon le marché" et "Exécution de bourse". En cas de l'exécution partielle l'ordre de marché ou limité avec le volume résiduel n'est pas remis et continue de fonctionner. Un ordre correspondant limité ORDER_TYPE_BUY_LIMIT/ ORDER_TYPE_SELL_LIMIT avec le type de l'exécution ORDER_FILLING_RETURN sera créé à l'activation pour les ordres ORDER_TYPE_BUY_STOP_LIMIT et ORDER_TYPE_SELL_STOP_LIMIT.	OrderGetInteger, HistoryOrderGetInteger

© 2000-2015, MetaQuotes Software Corp.

ORDER_MAGIC	L'identificateur de l'expert exposant l'ordre (est destiné pour que chaque expert expose le numéro personnel unique)	OrderGetInteger, HistoryOrderGetInteger
ORDER_POSITION_ID	L'identificateur de la position, qui est mis sur l'ordre à son exécution. Chaque ordre exécuté engendre le marché, qui ouvreune nouvelle postion ou change la position déjà existante. L'identificateur de cette position s'établit à l'ordre exécuté à ce moment.	OrderGetInteger, HistoryOrderGetInteger
ORDER_PRICE_CURRENT	Le prix actuel selon le symbole de l'ordre	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_OPEN	Le prix indiqué dans l'ordre	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_STOPLIMIT	Le prix de l'organisation de l'ordre à cours limité au fonctionnement de l'ordre StopLimit	OrderGetDouble, HistoryOrderGetDouble
ORDER_SL	Le niveau Stop Loss	OrderGetDouble, HistoryOrderGetDouble
ORDER_STATE	Le statut de l'ordre	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_CANCELED	L'ordre est retiré par le client	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_EXPIRED	L'ordre est retiré à l'expiration du terme ses actions	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_FILLED	L'ordre a été exécuté complètement	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PARTIAL	L'ordre a été exécuté partiellement	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PLACED	L'ordre est accepté	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REJECTED	L'ordre est rejeté	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_ADD	L'ordre dans l'état de l'enregistrement (l'exposition au système commercial)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_CAN	L'ordre dans l'état de la	OrderGetInteger,

© 2000-2015, MetaQuotes Software Corp.

CEL	suppression (la suppression du système commercial)	HistoryOrderGetInteger
ORDER_STATE_REQUEST_MODIFY	L'ordre dans l'état de la modification (le changement de ses paramètres)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_STARTED	L'ordre est contrôlé sur la convenance, mais n'est pas encore accepté par le broker	OrderGetInteger, HistoryOrderGetInteger
ORDER_SYMBOL	Le symbole, selon lequel on expose l'ordre	OrderGetString, HistoryOrderGetString
ORDER_TIME_DAY	L'ordre fonctionnera seulement pendant le jour commercial courant	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE	Le temps de l'exécution ou le retrait de l'ordre	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE_MSC	Le temps de l'exécution/du retrait de l'ordre en millisecondes depuis le 01.01.1970	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_EXPIRATION	Le temps de l'expiration de l'ordre	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_GTC	L'ordre se trouve au tour jusqu'à ce qu'il soit retiré	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP	Le temps de l'organisation de l'ordre	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP_MSC	Le temps de l'établissement de l'ordre pour l'exécution en millisecondes depuis le 01.01.1970	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED	L'ordre fonctionnera jusqu'à la date de l'expiration	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED_DAY	L'ordre sera valable jusqu'à 23:59:59 du jour indiqué. Si ce temps ne tombe pas sur la session commerciale, l'expiration se produira au temps commercial immédiat.	OrderGetInteger, HistoryOrderGetInteger
ORDER_TP	Le niveau Take Profit	OrderGetDouble, HistoryOrderGetDouble
ORDER_TYPE	Le type de l'ordre	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY	L'ordre de marché sur l'achat	OrderGetInteger,

© 2000-2015, MetaQuotes Software Corp.

		HistoryOrderGetInteger
ORDER_TYPE_BUY_LIMIT	L'ordre remis Buy Limit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP	L'ordre remis Buy Stop	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP_LIMIT	Pour l'obtention du prix de l'ordre se présente l'ordre remis Buy Limit au prix de StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_FILLING	Le type de l'exécution selon le reste	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL	L'ordre de marché sur la vente	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_LIMIT	L'ordre remis Sell Limit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP	L'ordre remis Sell Stop	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP_LIMIT	Pour l'obtention du prix de l'ordre se présente l'ordre remis Sell Limit au prix de StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_TIME	Le temps de la vie de l'ordre	OrderGetInteger, HistoryOrderGetInteger
ORDER_VOLUME_CURRENT	Le volume non exécuté	OrderGetDouble, HistoryOrderGetDouble
ORDER_VOLUME_INITIAL	Le volume initial à l'organisation de l'ordre	OrderGetDouble, HistoryOrderGetDouble
PERIOD_CURRENT	Période actuelle	Périodes des graphiques
PERIOD_D1	1 jour	Périodes des graphiques
PERIOD_H1	1 heure	Périodes des graphiques
PERIOD_H12	12 heures	Périodes des graphiques
PERIOD_H2	2 heures	Périodes des graphiques
PERIOD_H3	3 heures	Périodes des graphiques
PERIOD_H4	4 heures	Périodes des graphiques
PERIOD_H6	6 heures	Périodes des graphiques
PERIOD_H8	8 heures	Périodes des graphiques
PERIOD_M1	1 minute	Périodes des graphiques
PERIOD_M10	10 minutes	Périodes des graphiques

© 2000-2015, MetaQuotes Software Corp.

PERIOD_M12	12 minutes	Périodes des graphiques
PERIOD_M15	15 minutes	Périodes des graphiques
PERIOD_M2	2 minutes	Périodes des graphiques
PERIOD_M20	20 minutes	Périodes des graphiques
PERIOD_M3	3 minutes	Périodes des graphiques
PERIOD_M30	30 minutes	Périodes des graphiques
PERIOD_M4	4 minutes	Périodes des graphiques
PERIOD_M5	5 minutes	Périodes des graphiques
PERIOD_M6	6 minutes	Périodes des graphiques
PERIOD_MN1	1 mois	Périodes des graphiques
PERIOD_W1	1 semaine	Périodes des graphiques
PLOT_ARROW	Le code de la flèche pour le style DRAW_ARROW	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_ARROW_SHIFT	Le décalage des flèches selon la verticale pour le style DRAW_ARROW	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_COLOR_INDEXES	Le nombre de couleurs	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_BEGIN	Le nombre de barres initialles sans dessin et les valeurs dans DataWindow	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_TYPE	Le type de la construction graphique	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_EMPTY_VALUE	Une valeur vide pour la construction, pour laquelle il n'y a aucun dessin	PlotIndexSetDouble
PLOT_LABEL	Le nom de la série d'indicateur graphique pour l'affichage dans la fenêtre DataWindow. Pour les styles complexes graphiques demandant pour l'affichage quelques tampons d'indicateur, on peut spécifier les noms pour chaque tampon en utilisant comme délimiteur ";". L'exemple du code est dans DRAW_CANDLES	PlotIndexSetString
PLOT_LINE_COLOR	L'index du tampon qui contient la couleur du dessin	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_STYLE	Le style de la ligne du dessin	PlotIndexSetInteger,

© 2000-2015, MetaQuotes Software Corp.

		PlotIndexGetInteger
PLOT_LINE_WIDTH	L'épaisseur de la ligne du dessin	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHIFT	Le décalage de la construction graphique de l'indicateur selon l'axe du temps dans les barres	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHOW_DATA	Le critère de l'affichage des valeurs de la construction dans la fenêtre DataWindow	PlotIndexSetInteger, PlotIndexGetInteger
PLUSDI_LINE	La ligne +DI	Lignes des indicateurs
POINTER_AUTOMATIC	L'indicateur de n'importe quel objet créé automatiquement (sans utilisation new())	CheckPointer
POINTER_DYNAMIC	L'indicateur de l'objet créé par l'opérateur new	CheckPointer
POINTER_INVALID	L'indicateur incorrect	CheckPointer
POSITION_COMMENT	Le commentaire sur la position	PositionGetString
POSITION_COMMISSION	La commission	PositionGetDouble
POSITION_IDENTIFIER	L'identificateur de la position est un nombre unique, qui est assigné à chaque position ouverte à nouveau, et ne change pas pendant toute sa vie. Le virement de la position ne change pas l'identificateur de la position.	PositionGetInteger
POSITION_MAGIC	Magic number pour la position (regardes ORDER_MAGIC)	PositionGetInteger
POSITION_PRICE_CURRENT	La valeur présente selon le symbole	PositionGetDouble
POSITION_PRICE_OPEN	Le prix de la position	PositionGetDouble
POSITION_PROFIT	Un profit courant	PositionGetDouble
POSITION_SL	Le niveau Stop Loss pour une position ouverte	PositionGetDouble
POSITION_SWAP	Le swap accumulé	PositionGetDouble
POSITION_SYMBOL	Le symbole selon lequel la position est ouverte	PositionGetString
POSITION_TIME	Le temps de l'ouverture de la position	PositionGetInteger
POSITION_TIME_MSC	Le temps de l'ouverture de la	PositionGetInteger

© 2000-2015, MetaQuotes Software Corp.

	position en millisecondes depuis le 01.01.1970	
POSITION_TIME_UPDATE	Le temps du changement de la position en millisecondes depuis le 01.01.1970	PositionGetInteger
POSITION_TIME_UPDATE_MSC	Le temps du changement de la position en millisecondes depuis le 01.01.1970	PositionGetInteger
POSITION_TP	Le niveau Take Profit pour une position ouverte	PositionGetDouble
POSITION_TYPE	Le type de la position	PositionGetInteger
POSITION_TYPE_BUY	L'achat	PositionGetInteger
POSITION_TYPE_SELL	La vente	PositionGetInteger
POSITION_VOLUME	Le volume de la position	PositionGetDouble
PRICE_CLOSE	Le prix de la clôture	Constantes de prix
PRICE_HIGH	Le prix maximum pour la période	Constantes de prix
PRICE_LOW	Le prix minimum pour la période	Constantes de prix
PRICE_MEDIAN	Le prix moyen, (high+low)/2	Constantes de prix
PRICE_OPEN	Le prix de l'ouverture	Constantes de prix
PRICE_TYPICAL	Le prix typique, (high+low +close)/3	Constantes de prix
PRICE_WEIGHTED	Le prix moyen pondéré, (high +low+close+close)/4	Constantes de prix
PROGRAM_EXPERT	L'expert	MQLInfoInteger
PROGRAM_INDICATOR	L'indicateur	MQLInfoInteger
PROGRAM_SCRIPT	Le script	MQLInfoInteger
REASON_ACCOUNT	Un autre compte a été activé ou la reconnexion vers le serveur commercial est produite à la suite de changements dans les paramètres du compte	UninitializeReason, OnDeinit
REASON_CHARTCHANGE	Le caractère ou la période du graphique était changé	UninitializeReason, OnDeinit
REASON_CHARTCLOSE	Le graphique est fermé	UninitializeReason, OnDeinit
REASON_CLOSE	Le terminal a été fermé	UninitializeReason, OnDeinit

© 2000-2015, MetaQuotes Software Corp.

REASON_INITFAILED	Cette valeur signifit que OnInit() a rendu la valeur non nulle	UninitializeReason, OnDeinit
REASON_PARAMETERS	Les paramètres d'entrée étaient changés par l'utilisateur	UninitializeReason, OnDeinit
REASON_PROGRAM	L'expert a cessé le travail, ayant provoqué la fonction ExpertRemove()	UninitializeReason, OnDeinit
REASON_RECOMPILE	Le programme a été recompilé	UninitializeReason, OnDeinit
REASON_REMOVE	Le programme est supprimé du graphique	UninitializeReason, OnDeinit
REASON_TEMPLATE	Un autre modèle du graphique a été appliqué	UninitializeReason, OnDeinit
SATURDAY	Samedi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SEEK_CUR	La position courante de l'indicateur de fichier	FileSeek
SEEK_END	La fin du fichier	FileSeek
SEEK_SET	Le début du fichier	FileSeek
SENKOUSPANA_LINE	La ligne Senkou Span A	Lignes des indicateurs
SENKOUSPANB_LINE	La ligne Senkou Span B	Lignes des indicateurs
SERIES_BARS_COUNT	Le nombre de barres selon le symbole-la période pour ce moment	SeriesInfoInteger
SERIES_FIRSTDATE	La première date pour le symbole-la période pour le moment actuel	SeriesInfoInteger
SERIES_LASTBAR_DATE	Le temps de l'ouverture de la dernière barre selon le caractère-période	SeriesInfoInteger
SERIES_SERVER_FIRSTDATE	La première date à l'histoire selon le symbole sur le serveur indépendamment de la période	SeriesInfoInteger
SERIES_SYNCHRONIZED	Признак синхронизированности данных по символу/периоду на данный момент	SeriesInfoInteger
SERIES_TERMINAL_FIRSTDATE	La première date à l'histoire selon le symbole dans le	SeriesInfoInteger

© 2000-2015, MetaQuotes Software Corp.

	terminal de client indépendamment de la période	
SHORT_MAX	La valeur maximale, qui peut être présentée par le type short	Constantes des types de données numériques
SHORT_MIN	La valeur minimale, qui peut être présentée par le type short	Constantes des types de données numériques
SIGNAL_BASE_AUTHOR_LOGIN	Логин автора сигнала	SignalBaseGetString
SIGNAL_BASE_BALANCE	Баланс счета	SignalBaseGetDouble
SIGNAL_BASE_BROKER	Наименование брокера (компании)	SignalBaseGetString
SIGNAL_BASE_BROKER_SERVER	Сервер брокера	SignalBaseGetString
SIGNAL_BASE_CURRENCY	Валюта счета сигнала	SignalBaseGetString
SIGNAL_BASE_DATE_PUBLISHED	Дата публикации сигнала (когда стал доступен для подписки)	SignalBaseGetInteger
SIGNAL_BASE_DATE_STARTED	Дата начала мониторинга сигнала	SignalBaseGetInteger
SIGNAL_BASE_EQUITY	Средства на счете	SignalBaseGetDouble
SIGNAL_BASE_GAIN	Прирост счета в процентах	SignalBaseGetDouble
SIGNAL_BASE_ID	ID сигнала	SignalBaseGetInteger
SIGNAL_BASE_LEVERAGE	Плечо торгового счета	SignalBaseGetInteger
SIGNAL_BASE_MAX_DRAWDOWN	Максимальная просадка	SignalBaseGetDouble
SIGNAL_BASE_NAME	Имя сигнала	SignalBaseGetString
SIGNAL_BASE_PIPS	Результат торговли в пипсах	SignalBaseGetInteger
SIGNAL_BASE_PRICE	Цена подписки на сигнал	SignalBaseGetDouble
SIGNAL_BASE_RATING	Позиция в рейтинге сигналов	SignalBaseGetInteger
SIGNAL_BASE_ROI	Значение ROI (Return on Investment) сигнала в %	SignalBaseGetDouble
SIGNAL_BASE_SUBSCRIBERS	Количество подписчиков	SignalBaseGetInteger
SIGNAL_BASE_TRADE_MODE	Тип счета (0-реальный счет, 1-демо-счет, 2-конкурсный счет)	SignalBaseGetInteger
SIGNAL_BASE_TRADES	Количество трейдов	SignalBaseGetInteger

© 2000-2015, MetaQuotes Software Corp.

SIGNAL_INFO_CONFIRMATIONS_DISABLED	Флаг разрешения синхронизации без показа диалога подтверждения	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_COPY_SLTP	Флаг копирования Stop Loss и Take Profit	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_DEPOSIT_PERCENT	Ограничения по депозиту (в %)	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_EQUITY_LIMIT	Процент для конвертации объема сделки	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_ID	id сигнала, r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_NAME	Имя сигнала, r/o	SignalInfoGetString
SIGNAL_INFO_SLIPPAGE	Величина проскальзывания, с которым выставляются рыночные ордера при синхронизации позиций и копировании сделок	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_SUBSCRIPTION_ENABLED	Флаг разрешения на копирование сделок по подписке	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_TERMS_AGREE	Флаг согласия с условиями использования сервиса "Сигналы", r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_VOLUME_PERCENT	Значение ограничения по средствам для сигнала, r/o	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_LINE	La ligne de signal	Lignes des indicateurs
STAT_BALANCE_DD	Un prélèvement maximal de la balance en moyens monétaires. En train du commerce la balance peut éprouver la multitude de prélèvements, il faut prendre la plus grande signification.	TesterStatistics
STAT_BALANCE_DD_RELATIVE	Le prélèvement de la balance en moyens monétaires, qui a été fixé au moment du prélèvement maximal de la balance en pourcentage (STAT_BALANCE_DDREL_PERCENT).	TesterStatistics
STAT_BALANCE_DDREL_PERCENT	Un prélèvement maximal de la balance en pourcentage. En train du commerce la balance	TesterStatistics

© 2000-2015, MetaQuotes Software Corp.

	peut éprouver la multitude de prélèvements, on fixe une valeur relative en pourcentage et une valeur absolue monétaire pour chaque prélèvement. Rend la plus grande valeur	
STAT_BALANCEDD_PERCENT	Le prélèvement de la balance en pourcentage, qui a été fixé au moment du prélèvement maximal de la balance en moyens monétaires (STAT_BALANCE_DD).	TesterStatistics
STAT_BALANCEMIN	La valeur minimale de la balance	TesterStatistics
STAT_CONLOSSMAX	Une perte maximale au héritage des trades déficitaires. La valeur est moins ou égal au zéro	TesterStatistics
STAT_CONLOSSMAX_TRADES	Le nombre de trades qui ont forméSTAT_CONLOSSMAX (une perte maximale au héritage des trades déficitaires)	TesterStatistics
STAT_CONPROFITMAX	Un bénéfice maximal dans la plus grande valeur parmi toutes les trades profitables. La valeur est plus ou égal au zéro	TesterStatistics
STAT_CONPROFITMAX_TRADES	Le nombre de trades qui ont forméSTAT_CONPROFITMAX (un bénéfice maximal au héritage des trades profitables)	TesterStatistics
STAT_CUSTOM_ONTESTER	La valeur du critère calculé d'utilisateur de l'optimisation rendu par la fonctionOnTester()	TesterStatistics
STAT_DEALS	Le nombre de marchés faits	TesterStatistics
STAT_EQUITY_DD	Un prélèvement maximal des moyens en moyens monétaires. En train du commerce les moyens peuvent éprouver la multitude de prélèvements, il faut prendre	TesterStatistics

© 2000-2015, MetaQuotes Software Corp.

	la plus grande signification.	
STAT_EQUITY_DD_RELATIVE	Le prélèvement des moyens en moyens monétaires, qui a été fixé au moment du prélèvement maximal des moyens en pourcentage (STAT_EQUITY_DDREL_PERCENT).	TesterStatistics
STAT_EQUITY_DDREL_PERCENT	Un prélèvement maximal des moyens en pourcentage. En train du commerce les moyens peuvent éprouver la multitude de prélèvements, dont on fixe une valeur relative du prélèvement en porcentage. Rend la plus grande valeur	TesterStatistics
STAT_EQUITYDD_PERCENT	>Le prélèvement des moyens en pourcentage, qui a été fixé au moment du prélèvement maximal des moyens en moyens monétaires (STAT_EQUITY_DD).	TesterStatistics
STAT_EQUITYMIN	La valeur minimale des moyens propres	TesterStatistics
STAT_EXPECTED_PAYOFF	L'attente mathématique du gain	TesterStatistics
STAT_GROSS_LOSS	Une perte totale, la somme de tous les trades déficitaires (négatifs). La valeur est moins ou égal au zéro	TesterStatistics
STAT_GROSS_PROFIT	Le bénéfice total, la somme de tous les trades profitables (positifs). La valeur est plus ou égal au zéro	TesterStatistics
STAT_INITIAL_DEPOSIT	La valeur du dépôt initial	TesterStatistics
STAT_LONG_TRADES	Les trades longues	TesterStatistics
STAT_LOSS_TRADES	Les trades déficitaires	TesterStatistics
STAT_LOSSTRADES_AVGCON	Une moyenne longueur de la série déficitaire des trades	TesterStatistics
STAT_MAX_CONLOSS_TRADES	Le nombre de trades dans la plus longue série de trades déficitairesSTAT_MAX_CONLOSSES	TesterStatistics

© 2000-2015, MetaQuotes Software Corp.

STAT_MAX_CONLOSSES	La perte totale dans une plus longue série des trades déficitaires	TesterStatistics
STAT_MAX_CONPROFIT_TRADES	Le nombre de trades dans la plus longue série des trades profitablesSTAT_MAX_CONWINS	TesterStatistics
STAT_MAX_CONWINS	Le bénéfice total dans la plus longue série des trades profitables	TesterStatistics
STAT_MAX_LOSSTRADE	Une perte maximale - la valeur la plus petite parmi tous les trades déficitaires. La valeur est moins ou égal au zéro	TesterStatistics
STAT_MAX_PROFITTRADE	Un bénéfice maximal – la plus grande valeur parmi tous les trades profitables. La valeur est plus ou égal au zéro	TesterStatistics
STAT_MIN_MARGINLEVEL	La valeur minimale atteinte du niveau de la marge	TesterStatistics
STAT_PROFIT	Le bénéfice net après le test, la somme STAT_GROSS_PROFIT et STAT_GROSS_LOSS (STAT_GROSS_LOSS est toujours moins ou égal au zéro)	TesterStatistics
STAT_PROFIT_FACTOR	La rentabilité - le rapportSTAT_GROSS_PROFIT/ STAT_GROSS_LOSS. Si STAT_GROSS_LOSS=0, la rentabilité prend la valeurDBL_MAX	TesterStatistics
STAT_PROFIT_LONGTRADES	Les trades longues profitables	TesterStatistics
STAT_PROFIT_SHORTTRADES	Les trades courts profitables	TesterStatistics
STAT_PROFIT_TRADES	Les trades profitables	TesterStatistics
STAT_PROFITTRADES_AVGCON	Une moyenne longueur de la série profitable des trades	TesterStatistics
STAT_RECOVERY_FACTOR	Le facteur de la restitution – le rapportSTAT_PROFIT/ STAT_BALANCE_DD	TesterStatistics
STAT_SHARPE_RATIO	Le coefficient de Scharp	TesterStatistics

© 2000-2015, MetaQuotes Software Corp.

STAT_SHORT_TRADES	Les trades courts	TesterStatistics
STAT_TRADES	Le nombre de trades	TesterStatistics
STAT_WITHDRAWAL	Le nombre de moyens déduits du compte	TesterStatistics
STO_CLOSECLOSE	La construction aux prix Close/Close	Constantes de prix
STO_LOWHIGH	La construction aux prix Low/ High	Constantes de prix
STYLE_DASH	La ligne interrompue	Styles du dessin
STYLE_DASHDOT	La barre- la ligne pointillée	Styles du dessin
STYLE_DASHDOTDOT	La barre- deux points	Styles du dessin
STYLE_DOT	La ligne pointillée	Styles du dessin
STYLE_SOLID	La ligne continue	Styles du dessin
SUNDAY	Dimanche	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SYMBOL_ASK	Ask - une meilleure proposition pour l'achat	SymbolInfoDouble
SYMBOL_ASKHIGH	Ask maximum par jour	SymbolInfoDouble
SYMBOL_ASKLOW	Ask minimum par jour	SymbolInfoDouble
SYMBOL_BANK	La source de la cotation en cours	SymbolInfoString
SYMBOL_BASIS	Имя базового актива для производного инструмента	SymbolInfoString
SYMBOL_BID	Bid - une meilleure proposition pour la vente	SymbolInfoDouble
SYMBOL_BIDHIGH	Bid maximum par jour	SymbolInfoDouble
SYMBOL_BIDLOW	Bid minimum par jour	SymbolInfoDouble
SYMBOL_CALC_MODE_CFD	CFD mode – le calcul du gage et du profit pour CFD	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDINDEX	CFD index mode – le calcul du gage et du profit pour CFD par les index	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD Leverage mode – le calcul du gage et du profit pour CFD au commerce d'effet de levier	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES	Futures mode – le calcul de la garantie et des bénéfices pour	SymbolInfoInteger

© 2000-2015, MetaQuotes Software Corp.

	la commerce des contrats à terme sur la bourse	
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS Futures mode - le calcul de la garantie et des bénéfices pour la commerce des contrats à terme sur FORTS.	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_STOCKS	Exchange mode - le calcul de la garantie et des bénéfices pour la commerce des papiers monétaires sur la bourse	SymbolInfoInteger
SYMBOL_CALC_MODE_FOREX	Forex mode - le calcul du profit et la marge pour Forex	SymbolInfoInteger
SYMBOL_CALC_MODE_FUTURES	Futures mode - le calcul du gage et du profit pour les futures	SymbolInfoInteger
SYMBOL_CALC_MODE_SERV_COLLATERAL	Collateral mode - a symbol is used as a non-tradable asset on a trading account.	SymbolInfoInteger
SYMBOL_CURRENCY_BASE	La devise de base de l'instrument	SymbolInfoString
SYMBOL_CURRENCY_MARGIN	La devise des moyens hypothécaires	SymbolInfoString
SYMBOL_CURRENCY_PROFIT	La devise du bénéfice	SymbolInfoString
SYMBOL_DESCRIPTION	La description de ligne du caractère	SymbolInfoString
SYMBOL_DIGITS	Le nombre de signes après la virgule	SymbolInfoInteger
SYMBOL_EXPIRATION_DAY	L'ordre est valable jusqu'à la fin du jour	SymbolInfoInteger
SYMBOL_EXPIRATION_GTC	L'ordre est valable sans limitation par le temps jusqu'à son annulation évidente	SymbolInfoInteger
SYMBOL_EXPIRATION_MODE	Les drapeaux des modes permis de l'expiration de l'ordre	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED	Le délai de l'expiration est indiqué dans l'ordre	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED_DAY	Le jour de l'expiration est indiqué dans l'ordre	SymbolInfoInteger
SYMBOL_EXPIRATION_TIME	La date de l'achèvement des enchères selon l'instrument	SymbolInfoInteger

© 2000-2015, MetaQuotes Software Corp.

	(d'habitude il est utilisé pour les contrats à terme)	
SYMBOL_FILLING_FOK	Cette politique de l'exécution signifie que l'ordre peut être exceptionnellement exécuté dans le volume indiqué. Si au marché au moment donné il n'y a pas de volume suffisant de l'instrument financier, l'ordre ne sera pas exécuté. Le volume nécessaire peut être fait de quelques propositions accessibles au moment donné sur le marché.	SymbolInfoInteger
SYMBOL_FILLING_IOC	Dans ce cas le trader accepte de passer le marché par le volume au maximum accessible sur le marché dans la limite du volume indiqué dans l'ordre. En cas d'impossibilité de l'exécution complète l'ordre sera exécuté sur le volume accessible, et le volume non exécuté de l'ordre sera annulé. La possibilité de l'utilisation des ordres IOC est définie sur le serveur commercial.	SymbolInfoInteger
SYMBOL_FILLING_MODE	Les drapeaux des modes permis du remplissage de l'ordre	SymbolInfoInteger
SYMBOL_ISIN	Le nom du symbole commercial dans le système des codes internationaux identificatoires des valeurs mobilières— ISIN (International Securities Identification Number). Le numéro international d'identification de la valeur mobilière - c'est un code alphanumérique à 12 chiffres qui identifie la valeur mobilière absolument. La présence de la propriété donnée du symbole est définie au côté du serveur commercial.	SymbolInfoString

© 2000-2015, MetaQuotes Software Corp.

SYMBOL_LAST	Prix de la dernière affaire	SymbolInfoDouble
SYMBOL_LASTHIGH	Last maximum par jour	SymbolInfoDouble
SYMBOL_LASTLOW	Last minimum par jour	SymbolInfoDouble
SYMBOL_MARGIN_INITIAL	La marge initiale (initiante) désigne le montant des moyens nécessaires hypothécaires en devise de marge pour l'ouverture de la position par le volume qui correspond à un lot. Elle est utilisée au contrôle des moyens du client à l'entrée au marché.	SymbolInfoDouble
SYMBOL_MARGIN_MAINTENANCE	Le marge de maintenance selon l'instrument. En cas si elle est spécifiée – on indique le montant de la marge en devise de marge de l'instrument qui est retenue d'un lot. Elle est utilisé au contrôle des moyens du client au changement de l'état du compte du client. Si la marge de maintenance est égale au 0, on utilise la marge initiale.	SymbolInfoDouble
SYMBOL_OPTION_MODE	Тип опциона	SymbolInfoInteger
SYMBOL_OPTION_MODE_AMERICAN	Европейский тип опциона – может быть погашен только в указанную дату (дата истечения срока, дата исполнения, дата погашения)	SymbolInfoInteger
SYMBOL_OPTION_MODE_EUROPEAN	Американский тип опциона – может быть погашен в любой день до истечения срока опциона. Для такого типа задаётся период, в течение которого покупатель может исполнить данный опцион	SymbolInfoInteger
SYMBOL_OPTION_RIGHT	Право опциона (Call/Put)	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_CALL	Опцион, дающий право купить актив по фиксированной цене	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_PUT	Опцион, дающий право продать актив по фиксированной цене	SymbolInfoInteger

© 2000-2015, MetaQuotes Software Corp.

SYMBOL_OPTION_STRIKE	Цена исполнения опциона. Это цена, по которой покупатель опциона может купить (при опционе Call) или продать (при опционе Put) базовый актив, а продавец опциона соответственно обязан продать или купить соответствующее количество базового актива.	SymbolInfoDouble
SYMBOL_ORDER_LIMIT	Les ordres limites sont permis (Buy Limit et Sell Limit)	SymbolInfoInteger
SYMBOL_ORDER_MARKET	Les ordres de marché sont permis (Buy et Sell)	SymbolInfoInteger
SYMBOL_ORDER_MODE	Les drapeaux des types permis de l'ordre	SymbolInfoInteger
SYMBOL_ORDER_SL	L'établissement de Stop Loss est permis	SymbolInfoInteger
SYMBOL_ORDER_STOP	Les ordres stop sont permis (Buy Stop et Sell Stop)	SymbolInfoInteger
SYMBOL_ORDER_STOP_LIMIT	Les ordres stop limites sont permis (Buy Stop Limit et Sell Stop Limit)	SymbolInfoInteger
SYMBOL_ORDER_TP	L'établissement de Take Profit est permis	SymbolInfoInteger
SYMBOL_PATH	Le chemin dans l'arbre des caractères	SymbolInfoString
SYMBOL_POINT	La valeur de point de symbole	SymbolInfoDouble
SYMBOL_SELECT	Le symbole est choisi dans Market Watch	SymbolInfoInteger
SYMBOL_SESSION_AW	Le prix moyen pondéré de la session	SymbolInfoDouble
SYMBOL_SESSION_BUY_ORDERS	Le nombre total d'ordres à l'achat pour le moment	SymbolInfoInteger
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Le volume total des ordres à l'achat pour le moment	SymbolInfoDouble
SYMBOL_SESSION_CLOSE	Le prix de la clôture de la session	SymbolInfoDouble
SYMBOL_SESSION_DEALS	Le nombre de marchés dans la session courante	SymbolInfoInteger
SYMBOL_SESSION_INTEREST	Le volume total des positions	SymbolInfoDouble

© 2000-2015, MetaQuotes Software Corp.

	ouvertes	
SYMBOL_SESSION_OPEN	Le prix de l'ouverture de la session	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MAX	La valeur au maximum admissible du prix de la session	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MIN	La valeur au minimum admissible du prix de la session	SymbolInfoDouble
SYMBOL_SESSION_PRICE_SETTLEMENT	Le prix de la livraison à la session courante	SymbolInfoDouble
SYMBOL_SESSION_SELL_ORDERS	Le nombre total d'ordres à la vente pour le moment	SymbolInfoInteger
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Le volume total des ordres à la vente pour le moment	SymbolInfoDouble
SYMBOL_SESSION_TURNOVER	Le chiffre d'affaires total des marchés à la session courante	SymbolInfoDouble
SYMBOL_SESSION_VOLUME	Le volume total des marchés à la session courante	SymbolInfoDouble
SYMBOL_SPREAD	La valeur de spread dans les points	SymbolInfoInteger
SYMBOL_SPREAD_FLOAT	L'indication du spread flottant	SymbolInfoInteger
SYMBOL_START_TIME	La date du commencement des enchères selon l'instrument (d'habitude il est utilisé pour les contrats à terme)	SymbolInfoInteger
SYMBOL_SWAP_LONG	La valeur de swap à l'achat	SymbolInfoDouble
SYMBOL_SWAP_MODE	Le modèle de calcul de swap	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Les swaps sont calculés dans l'argent en devise du dépôt du client	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Les swaps sont calculés dans l'argent en devise de marge du symbole	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Les swaps sont calculés dans l'argent en devise de base du symbole	SymbolInfoInteger
SYMBOL_SWAP_MODE_DISABLED	Pas de swap	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTERE	Les swaps sont calculés dans	SymbolInfoInteger

© 2000-2015, MetaQuotes Software Corp.

ST_CURRENT	les pour-cent annuels du prix de l'instrument au moment du calcul du swap (le mode bancaire – 360 jours en année)	
SYMBOL_SWAP_MODE_INTEREST_OPEN	Les swaps sont calculés dans les pour-cent annuels du prix de la position de l'ouverture selon le symbole (le mode bancaire – 360 jours en année)	SymbolInfoInteger
SYMBOL_SWAP_MODE_POINTS	Les swaps sont calculés dans les points	SymbolInfoInteger
SYMBOL_SWAP_MODE_REOPEN_BID	Les swaps sont calculés par l'ouverture répétée de la position. A la fin du jour commercial la position se ferme forcément. Le lendemain la position s'ouvre de nouveau selon le prix courant Bid +/- le nombre indiqué de points (dans les paramètres SYMBOL_SWAP_LONG et SYMBOL_SWAP_SHORT)	SymbolInfoInteger
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Les swaps sont calculés par l'ouverture répétée de la position. A la fin du jour commercial la position se ferme forcément. Le lendemain la position de l'ouverture répétée selon le prix de la clôture + / - le nombre indiqué de points (dans les paramètres SYMBOL_SWAP_LONG et SYMBOL_SWAP_SHORT)	SymbolInfoInteger
SYMBOL_SWAP_ROLLOVER3DAYS	Le jour de la semaine pour le calcul de swap triple	SymbolInfoInteger
SYMBOL_SWAP_SHORT	La valeur de swap à la vente	SymbolInfoDouble
SYMBOL_TICKS_BOOKDEPTH	Le nombre maximal des demandes montrées au profondeur de marché. Pour les instruments qui n'ont pas de tour des demandes, la valeur est égal au 0	SymbolInfoInteger
SYMBOL_TIME	Le temps de la dernière cotation	SymbolInfoInteger

© 2000-2015, MetaQuotes Software Corp.

SYMBOL_TRADE_CALC_MODE	La mode de calcul des prix de contrat	SymbolInfoInteger
SYMBOL_TRADE_CONTRACT_SIZE	Le montant du contrat commercial	SymbolInfoDouble
SYMBOL_TRADE_EXECUTION_EXCHANGE	L'exécution boursière	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_INSTANT	Le commerce sur prix de courant	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_MARKET	L'exécution des ordres selon le marché	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_REQUEST	Le commerce sur la demande	SymbolInfoInteger
SYMBOL_TRADE_EXEMODE	La mode d'exécution d'affaire	SymbolInfoInteger
SYMBOL_TRADE_FREEZE_LEVEL	La distance pour bloquer des opérations commerciales (dans les points)	SymbolInfoInteger
SYMBOL_TRADE_MODE	Le type d'exécution d'ordre	SymbolInfoInteger
SYMBOL_TRADE_MODE_CLOSEONLY	On permet seulement les opérations de la clôture des positions	SymbolInfoInteger
SYMBOL_TRADE_MODE_DISABLED	Le commerce selon le symbole est interdit	SymbolInfoInteger
SYMBOL_TRADE_MODE_FULL	Il n'y a pas de limitations sur les transactions commerciales	SymbolInfoInteger
SYMBOL_TRADE_MODE_LONGONLY	On permet seulement les achats	SymbolInfoInteger
SYMBOL_TRADE_MODE_SHORTONLY	On permet seulement les ventes	SymbolInfoInteger
SYMBOL_TRADE_STOPS_LEVEL	L'alinéa minimal dans les points du prix courant de la clôture pour l'installation des ordres Stop	SymbolInfoInteger
SYMBOL_TRADE_TICK_SIZE	Le changement minimal du prix	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE	La valeur SYMBOL_TRADE_TICK_VALUE_PROFIT	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_LOSS	Le coût calculé du tick pour la position déficitaire	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_	Le coût calculé du tick pour la	SymbolInfoDouble

© 2000-2015, MetaQuotes Software Corp.

PROFIT	position profitable	
SYMBOL_VOLUME	Le volume - le volume dans le dernier marché	SymbolInfoInteger
SYMBOL_VOLUME_LIMIT	Un volume cumulé au maximum admissible pour le symbole donné de la position ouverte et des ordres remis dans une direction (l'achat ou la vente). Par exemple, à la limitation aux 5 lots on peut avoir la position ouverte sur l'achat par le volume de 5 lots et exposer l'ordre remis Sell Limit par le volume de 5 lots. Mais on ne peut pas exposer l'ordre remis Buy Limit (puisque un volume cumulé dans une direction excédera la limitation) ou exposer Sell Limit par le volume plus de 5 lots.	SymbolInfoDouble
SYMBOL_VOLUME_MAX	Le volume maximal pour une affaire	SymbolInfoDouble
SYMBOL_VOLUME_MIN	Le volume minimal pour une affaire	SymbolInfoDouble
SYMBOL_VOLUME_STEP	Un pas minimal du changement du volume pour une affaire	SymbolInfoDouble
SYMBOL_VOLUMEHIGH	Le volume maximum par jour	SymbolInfoInteger
SYMBOL_VOLUMELOW	Le volume minimum par jour	SymbolInfoInteger
TENKANSEN_LINE	La ligne Tenkan-sen	Lignes des indicateurs
TERMINAL_BUILD	Le numéro de build du terminal lancé	TerminalInfoInteger
TERMINAL_CODEPAGE	Le numéro de la page de code du langage est mis dans le terminal de client	TerminalInfoInteger
TERMINAL_COMMONDATA_PATH	Le dossier commun pour tous les terminals de client, installés sur un ordinateur	TerminalInfoString
TERMINAL_COMMUNITY_ACCOUNT	Флаг наличия авторизационных данных MQL5.community в терминале	TerminalInfoInteger
TERMINAL_COMMUNITY_BALA	Баланс пользователя в	TerminalInfoDouble

© 2000-2015, MetaQuotes Software Corp.

NCE	MQL5.community	
TERMINAL_COMMUNITY_CONNECTION	Наличие подключения к MQL5.community	TerminalInfoInteger
TERMINAL_COMPANY	Le nom de la compagnie	TerminalInfoString
TERMINAL_CONNECTED	La présence de la connexion au serveur commercial	TerminalInfoInteger
TERMINAL_CPU_CORES	Le nombre de processeurs dans le système	TerminalInfoInteger
TERMINAL_DATA_PATH	Le dossier où se trouvent les données du terminal	TerminalInfoString
TERMINAL_DISK_SPACE	La quantité d'espace disque libre pour le dossier MQL5\Files du terminal (agent) en Mb	TerminalInfoInteger
TERMINAL_DLLS_ALLOWED	La permission de l'utilisation DLL	TerminalInfoInteger
TERMINAL_EMAIL_ENABLED	La permission de l'expédition des lettres avec l'utilisation du serveur SMTP et le nom d'utilisateur, indiqué dans les réglages du terminal	TerminalInfoInteger
TERMINAL_FTP_ENABLED	La permission de l'expédition des rapports par FTP sur le serveur indiqué pour le terminal indiqué dans les réglages du compte commercial	TerminalInfoInteger
TERMINAL_LANGUAGE	Langage du terminal	TerminalInfoString
TERMINAL_MAXBARS	Le nombre maximum des barres sur le graphique	TerminalInfoInteger
TERMINAL_MEMORY_AVAILABLE	La taille de la mémoire libre du procès du terminal (agent) en Mb	TerminalInfoInteger
TERMINAL_MEMORY_PHYSICAL	La taille de la mémoire physique dans le système, en Mb	TerminalInfoInteger
TERMINAL_MEMORY_TOTAL	La taille de la mémoire disponible au procès du terminal (agent), en Mb	TerminalInfoInteger
TERMINAL_MEMORY_USED	La taille de la mémoire utilisée par le terminal (agent) en Mb	TerminalInfoInteger

© 2000-2015, MetaQuotes Software Corp.

TERMINAL_MQID	Флаг наличия MetaQuotes ID для отправки Push-уведомлений	TerminalInfoInteger
TERMINAL_NAME	Le nom du terminal	TerminalInfoString
TERMINAL_NOTIFICATIONS_ENABLED	Разрешение на отправку уведомлений на смартфон	TerminalInfoInteger
TERMINAL_OPENCL_SUPPORT	La version de l' OpenCL soutenu dans l'aspect 0x00010002 = 1.2. "0" signifit, que OpenCL n'est pas soutenu	TerminalInfoInteger
TERMINAL_PATH	Le dossier d'où le terminal a été démarré	TerminalInfoString
TERMINAL_TRADE_ALLOWED	La permission au commerce	TerminalInfoInteger
TERMINAL_X64	Le critère "le terminal de 64 bits"	TerminalInfoInteger
THURSDAY	Jeudi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TRADE_ACTION_DEAL	Placer l'ordre commercial pour l'exécution immédiate avec les paramètres indiqués (l'ordre du marché)	MqlTradeRequest
TRADE_ACTION_MODIFY	Modifiez les paramètres de l'ordre placé auparavant	MqlTradeRequest
TRADE_ACTION_PENDING	Placer l'ordre commercial pour l'exécution sous les conditions indiquées (l'ordre remis)	MqlTradeRequest
TRADE_ACTION_REMOVE	Supprimer l'ordrecommercial auparavant remis	MqlTradeRequest
TRADE_ACTION_SLTP	Changer les valeurs Stop Loss et Take Profit de la position ouverte	MqlTradeRequest
TRADE_RETCODE_CANCEL	La requête est annulé par un broker	MqlTradeResult
TRADE_RETCODE_CLIENT_DISABLES_AT	L'autotrading est interdit par le terminal de client	MqlTradeResult
TRADE_RETCODE_CONNECTION	Il n'y a pas de connection avec le serveur commercial.	MqlTradeResult
TRADE_RETCODE_DONE	La requête est accomplie	MqlTradeResult
TRADE_RETCODE_DONE_PARTI	La requête est partiellement	MqlTradeResult

© 2000-2015, MetaQuotes Software Corp.

AL	accomplie	
TRADE_RETCODE_ERROR	L'erreur de traitement de requête	MqlTradeResult
TRADE_RETCODE_FROZEN	L'ordre ou la position se sont gelés	MqlTradeResult
TRADE_RETCODE_INVALID	La requête incorrecte	MqlTradeResult
TRADE_RETCODE_INVALID_EXPIRATION	La date incorrecte de l'expiration de l'ordre dans la requête	MqlTradeResult
TRADE_RETCODE_INVALID_FILL	On indique le type non soutenu de l'exécution de l'ordre selon le reste	MqlTradeResult
TRADE_RETCODE_INVALID_ORDER	le type de l'ordreincorrecte ou interdit	MqlTradeResult
TRADE_RETCODE_INVALID_PRICE	Le prix incorrect dans la requête	MqlTradeResult
TRADE_RETCODE_INVALID_STOPS	Les Stops incorrects dans la requête	MqlTradeResult
TRADE_RETCODE_INVALID_VOLUME	Le volume incorrect dans la requête	MqlTradeResult
TRADE_RETCODE_LIMIT_ORDERS	La limite sur la quantité d'ordres remis est atteint	MqlTradeResult
TRADE_RETCODE_LIMIT_VOLUME	La limite sur le volume des ordres et les positions pour le symbole donné est atteint	MqlTradeResult
TRADE_RETCODE_LOCKED	La requête est bloquée pour le traitement	MqlTradeResult
TRADE_RETCODE_MARKET_CLOSED	Le marché est fermé	MqlTradeResult
TRADE_RETCODE_NO_CHANGES	Il n'y a pas de changements dans la requête	MqlTradeResult
TRADE_RETCODE_NO_MONEY	Il n'y a pas de ressources suffisantes pour l'exécution de la requête	MqlTradeResult
TRADE_RETCODE_ONLY_REAL	L'opération est permise seulement pour les comptes réels	MqlTradeResult
TRADE_RETCODE_ORDER_CHANGED	L'état de l'ordre a changé	MqlTradeResult
TRADE_RETCODE_PLACED	L'ordre a placé	MqlTradeResult

© 2000-2015, MetaQuotes Software Corp.

TRADE_RETCODE_POSITION_CLOSED	La position avecPOSITION_IDENTIFIERindiquée est déjà fermée	MqlTradeResult
TRADE_RETCODE_PRICE_CHANGED	Les prix ont changé	MqlTradeResult
TRADE_RETCODE_PRICE_OFF	Les cotations pour le traitement de la requête manquent	MqlTradeResult
TRADE_RETCODE_REJECT	La requête rejetée	MqlTradeResult
TRADE_RETCODE_REQUOTE	La recitation	MqlTradeResult
TRADE_RETCODE_SERVER_DISABLES_AT	L'autotrading est interdit par le serveur	MqlTradeResult
TRADE_RETCODE_TIMEOUT	La requête est supprimée à l'expiration du temps	MqlTradeResult
TRADE_RETCODE_TOO_MANY_REQUESTS	Les requêtes trop fréquentes	MqlTradeResult
TRADE_RETCODE_TRADE_DISABLED	Le commerce est interdit	MqlTradeResult
TRADE_TRANSACTION_DEAL_ADD	L'ajout du marché dans l'histoire. Se réalise à la suite de l'exécution de l'ordre ou de la tenue des opérations avec la balance du compte.	MqlTradeTransaction
TRADE_TRANSACTION_DEAL_DELETE	La suppression du marché de l'histoire. Les situations sont possibles, quand le marché auparavant exécuté est supprimé sur le serveur. Par exemple, le marché était supprimé dans le système (la bourse) extérieur commercial, où il était déduit par le broker.	MqlTradeTransaction
TRADE_TRANSACTION_DEAL_UPDATE	Le changement du marché dans l'histoire. Les situations sont possibles, quand le marché auparavant exécuté se change sur le serveur. Par exemple, le marché était changé dans le système (la bourse) extérieur commercial, où il était déduit par le broker.	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_ADD	L'ajout de l'ordre dans l'histoire à la suite de l'exécution ou de la	MqlTradeTransaction

© 2000-2015, MetaQuotes Software Corp.

	suppression.	
TRADE_TRANSACTION_HISTORY_DELETE	La suppression de l'ordre de l'histoire des ordres. Ce type est prévu pour l'élargissement de la fonctionnalité sur le côté du serveur commercial.	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_UPDATE	Le changement de l'ordre qui se trouve dans l'histoire des ordres. Ce type est prévu pour l'élargissement de la fonctionnalité sur le côté du serveur commercial.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_ADD	L'ajout d'un nouvel ordre ouvert.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_DELETE	La suppression de l'ordre de la liste des ouverts. L'ordre peut être supprimé des ouverts à la suite de l'exposition de la demande correspondante ou à la suite de l'exécution (du remplissage) et le transfert dans l'histoire.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_UPDATE	Le changement de l'ordre ouvert. Non seulement les changements évidents du côté du terminal de client ou du serveur commercial se rapportent aux changements donnés, mais aussi le changement de son état à son exposition (par exemple, le passage de l'étatORDER_STATE_STARTED au ORDER_STATE_PLACED ou de ORDER_STATE_PLACED au ORDER_STATE_PARTIAL etc.).	MqlTradeTransaction
TRADE_TRANSACTION_POSITION	Le changement de la position non lié à l'exécution du marché. Ce type de la transaction témoigne notamment de ce que la position était changée sur le côté du serveur commercial. Le volume, le prix de l'ouverture, ainsi que les niveaux Stop Loss et Take Profit peuvent être changés	MqlTradeTransaction

© 2000-2015, MetaQuotes Software Corp.

	dans la position. L'information sur les changements est transmise dans la structure MqlTradeTransaction par le gestionnaire OnTradeTransaction. Le changement de la position (l'ajout, le changement ou la liquidation) à la suite de l'exécution du marché n'entraîne pas l'apparition de la transaction TRADE_TRANSACTION_POSITION.	
TRADE_TRANSACTION_REQUEST	La notification que la demande commerciale est traitée par le serveur, et le résultat de son traitement est reçu. Pour les transaction de ce type dans la structureMqlTradeTransaction il faut analyser seulement un champ - type (le type de la transaction). Pour la réception de l'information supplémentaire il est nécessaire d'analyser les deuxièmes et troisièmes paramètres de la fonction OnTradeTransaction (request et result).	MqlTradeTransaction
TUESDAY	Mardi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TYPE_BOOL	bool	MqlParam
TYPE_CHAR	char	MqlParam
TYPE_COLOR	color	MqlParam
TYPE_DATETIME	datetime	MqlParam
TYPE_DOUBLE	double	MqlParam
TYPE_FLOAT	float	MqlParam
TYPE_INT	int	MqlParam
TYPE_LONG	long	MqlParam
TYPE_SHORT	short	MqlParam
TYPE_STRING	string	MqlParam

© 2000-2015, MetaQuotes Software Corp.

TYPE_UCHAR	uchar	MqlParam
TYPE_UINT	uint	MqlParam
TYPE_ULONG	ulong	MqlParam
TYPE_USHORT	ushort	MqlParam
UCHAR_MAX	La valeur maximale, qui peut être présentée par le type uchar	Constantes des types de données numériques
UINT_MAX	La valeur maximale, qui peut être présentée par le type uint	Constantes des types de données numériques
ULONG_MAX	La valeur maximale, qui peut être présentée par le type ulong	Constantes des types de données numériques
UPPER_BAND	Une limite supérieure	Lignes des indicateurs
UPPER_HISTOGRAM	L'histogramme supérieur	Lignes des indicateurs
UPPER_LINE	Une ligne supérieure	Lignes des indicateurs
USHORT_MAX	La valeur maximale, qui peut être présentée par le type ushort	Constantes des types de données numériques
VOLUME_REAL	Le volume commercial	Constantes de prix
VOLUME_TICK	Le volume de tick	Constantes de prix
WEDNESDAY	Mercredi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
WHOLE_ARRAY	Signifie le nombre d'éléments restés jusqu'à la fin du tableau, c'est-à-dire, tout le tableau sera traité	Les autres constantes
WRONG_VALUE	La constante peut non évidemment être amenée au type de n'importe quelle énumération	Les autres constantes

© 2000-2015, MetaQuotes Software Corp.