

Язык программирования

MQL5:



**Продвинутое использование
торговой платформы
MetaTrader 5**

Тимур Машнин

**Язык программирования
MQL5: Продвинутое
использование торговой
платформы MetaTrader 5**

«Издательские решения»

Машнин Т. С.

Язык программирования MQL5: Продвинутое
использование торговой платформы MetaTrader 5 /
Т. С. Машнин — «Издательские решения»,

Создание пользовательских индикаторов и советников для торговой
платформы MetaTrader 5 с использованием языка программирования
MQL5.

Содержание

Введение	6
Общая структура индикатора	7
Свойства индикатора	8
Параметры ввода и переменные индикатора	19
Хэндл индикатора	23
Функции OnInit (), OnDeinit (), OnCalculate ()	27
Функция OnInit ()	28
Функция OnDeinit ()	38
Функция OnCalculate ()	39
Пример создания индикатора	47
Графические объекты	58
Функция PlaySound	66
Функция OnChartEvent	70
Объектно-ориентированный подход	75
Общая структура советника	82
Функция OnTick ()	84
Пример создания эксперта	103
Пример создания эксперта с использованием ООП	135
Тестирование советников	156
Управление капиталом и оценка эффективности эксперта	164
Создание эксперта с помощью мастера MQL5 Wizard	184
Создание индикатора на основе модулей торговых сигналов эксперта	200
Генетические алгоритмы	210

Язык программирования MQL5: Продвинутое использование торговой платформы MetaTrader 5 Тимур Машнин

© Тимур Машнин, 2016

© Тимур Машнин, дизайн обложки, 2016

Создано в интеллектуальной издательской системе Ridero

Введение

Надеюсь, вы все уже прочитали справочник MQL5 на сайте <https://www.mql5.com/ru/docs>.

Здесь мы не будем пересказывать этот документ, а сосредоточимся на его практическом использовании. Мы будем позволять себе изредка только его цитирование.

Как сказано в предисловии к справочнику:

Программы, написанные на MetaQuotes Language 5, имеют различные свойства и предназначение

И далее идет перечисление Советник, Пользовательский индикатор, Скрипт, Библиотека и Включаемый файл.

Скрипты используются для выполнения одноразовых действий, обрабатывая только событие своего запуска, и поэтому не будут нам здесь интересны.

Также нам не будут интересны библиотеки, так как использование включаемых файлов более предпочтительно для уменьшения накладных расходов.

Поэтому мы сосредоточимся на создании советников и индикаторов с использованием включаемых файлов. Такова наша цель применения языка программирования MQL5, синтаксис которого конечно интересен, но будет нам только в помощь.

На самом деле программирование на языке MQL5 представляет собой яркий пример событийно-ориентированного программирования, так как весь код MQL5-приложения построен на переопределении функций обратного вызова – обработчиков событий клиентского терминала и пользователя. А уже в коде функций обратного вызова можно использовать либо процедурное программирование, либо объектно-ориентированное программирование. Здесь мы рассмотрим оба этих подхода.

Общая структура индикатора

Код индикатора начинается с блока объявления свойств индикатора и различных объектов, используемых индикатором, таких как массивы буферов индикатора, параметры ввода, глобальные переменные, хэндлы используемых технических индикаторов, константы.

Данный блок кода выполняется приложением Торговая Платформа MetaTrader 5 сразу при присоединении индикатора к графику символа.

После блока объявления свойств индикатора, его параметров и переменных, идет описание функций обратного вызова, которые терминал вызывает при наступлении таких событий, как инициализация индикатора после его загрузки, перед деинициализацией индикатора, при изменении ценовых данных, при изменении графика символа пользователем.

Для обработки вышеуказанных событий необходимо описать такие функции как `OnInit ()`, `OnDeinit ()`, `OnCalculate ()` и `OnChartEvent ()`.

В функции `OnInit ()` индикатора, как правило, объявленные в начальном блоке массивы связываются с буферами индикатора, определяя его выводимые значения, задаются цвета индикатора, точность отображения значений индикатора, его подписи и другие параметры отображения индикатора. Кроме того, в функции `OnInit ()` индикатора могут получаться хэндлы используемых технических индикаторов и рассчитываться другие используемые переменные.

В функции `OnDeinit ()` индикатора, как правило, с графика символа удаляются графические объекты индикатора, а также удаляются хэндлы используемых технических индикаторов.

В функции `OnCalculate ()` собственно и производится расчет значений индикатора, заполняя ими объявленные в начальном блоке массивы, которые в функции `OnInit ()` индикатора были связаны с буферами индикатора, данные из которых берутся терминалом для отрисовки индикатора. Кроме того, в функции `OnCalculate ()` могут изменяться цвета индикатора и другие параметры его отображения.

В функции `OnChartEvent ()` могут обрабатываться события, генерируемые другими индикаторами на графике, а также удаление пользователем графического объекта индикатора и другие события, возникающие при работе пользователя с графиком.

На этом код индикатора заканчивается, хотя там могут быть также определены пользовательские функции, которые вызываются из функций обратного вызова `OnInit ()`, `OnDeinit ()`, `OnCalculate ()` и `OnChartEvent ()`.

Свойства индикатора

Цитата из справочника:

Свойства программ (#property). У каждой mql5-программы можно указать дополнительные специфические параметры #property, которые помогают клиентскому терминалу правильно обслуживать программы без необходимости их явного запуска. В первую очередь это касается внешних настроек индикаторов. Свойства, описанные во включаемых файлах, полностью игнорируются. Свойства необходимо задавать в главном mq5-файле. #property идентификатор значение

В качестве первого свойства, как правило, указывается имя разработчика, например:

```
#property copyright «2009, MetaQuotes Software Corp.»
```

Далее указывается ссылка на сайт разработчика:

```
#property link http://www.mql5.com
```

После этого идет описание индикатора, каждая строка которого обозначается с помощью идентификатора description, например:

```
#property description «Average Directional Movement Index»
```

Далее указывается версия индикатора:

```
#property version «1.00»
```

На этом, как правило, объявление общих свойств индикатора заканчивается.

Индикатор может появляться в окне терминала двумя способами – на графике символа или в отдельном окне под графиком символа.

Свойство:

```
#property indicator_chart_window
```

Определяет отрисовку индикатора на графике символа.

А свойство:

```
#property indicator_separate_window
```

Определяет вывод индикатора в отдельное окно.

Одно из самых важных свойств индикатора – это количество буферов для расчета индикатора, например:

```
#property indicator_buffers 6
```

Данное свойство тесно связано с двумя другими свойствами индикатора – количеством графических построений и видом графических построений.

Количество графических построений это количество цветных диаграмм, составляющих индикатор.

Например, для индикатора ADX:

```
#property indicator_plots 3
```

Индикатор состоит из трех диаграмм (линий) – индикатора направленности +DI, индикатора направленности —DI и самого индикатора ADX.

Вид графических построений – это та графическая форма, из которой составляется график индикатора.

Например, для индикатора ADX:

```
#property indicator_type1 DRAW_LINE
```

```
#property indicator_type2 DRAW_LINE
```

```
#property indicator_type3 DRAW_LINE
```

Таким образом, каждая диаграмма индикатора ADX – это линия.

Графическая форма сопоставляется с графическим построением с помощью номера графического построения, следующего после indicator_type.

Цвет каждого графического построения индикатора задается свойством `indicator_colorN`.

Например, для индикатора ADX:

```
#property indicator_color1 LightSeaGreen  
#property indicator_color2 YellowGreen  
#property indicator_color3 Wheat
```

Цвет сопоставляется с графическим построением с помощью номера графического построения, следующего после `indicator_color`.

В справочнике есть таблица Web-цветов для определения цвета графического построения.

Вернемся к количеству буферов для расчета индикатора.

Так как данные для построения каждой диаграммы индикатора берутся из своего буфера индикатора, количество заявленных буферов индикатора не может быть меньше, чем заявленное число графических построений индикатора.

Сразу же возникает вопрос, каким образом конкретный массив, представляющий буфер индикатора, сопоставляется с конкретным графическим построением индикатора.

Делается это в функции обратного вызова `OnInit ()` с помощью вызова функции `SetIndexBuffer`.

Например, для индикатора ADX:

```
SetIndexBuffer (0,ExtADXBuffer);  
SetIndexBuffer (1,ExtPDIBuffer);  
SetIndexBuffer (2,ExtNDIBuffer);
```

Где первый аргумент, это номер графического построения.

Таким образом, массив связывается с диаграммой индикатора, а диаграмма связывается с ее формой и цветом.

Однако с буферами индикатора все немного сложнее.

Их количество может быть заявлено больше, чем количество графических построений индикатора.

Что это означает?

Это означает, что некоторые массивы, представляющие буфера индикатора, используются не для построения диаграмм индикатора, а для промежуточных вычислений.

Например, для индикатора ADX:

```
SetIndexBuffer (3,ExtPDBuffer, INDICATOR_CALCULATIONS);  
SetIndexBuffer (4,ExtNDBuffer, INDICATOR_CALCULATIONS);  
SetIndexBuffer (5,ExtTmpBuffer, INDICATOR_CALCULATIONS);
```

Такой массив определяется с помощью третьего параметра `INDICATOR_CALCULATIONS`.

Это дает следующее:

Все дело в частичном заполнении массива.

Если массив, указанный в функции `SetIndexBuffer`, является динамическим, т.е. объявлен без указания размера, но он привязан к буферу индикатора с помощью функции `SetIndexBuffer`, клиентский терминал сам заботится о том, чтобы размер такого массива соответствовал ценовой истории.

Рассмотрим это на примере индикатора ADX.

Откроем приложение MetaTrader 5 и в меню Tools (Сервис) выберем MetaQuotes Language Editor (Редактор MetaQuotes Language).

В редакторе MQL5, в окне Navigator (Навигатор), в разделе Indicators-> Examples выберем и откроем исходный код индикатора ADX.

В функции `OnInit ()` закомментируем строку:

```
// — indicator buffers
SetIndexBuffer (0,ExtADXBuffer);
SetIndexBuffer (1,ExtPDIBuffer);
SetIndexBuffer (2,ExtNDIBuffer);
SetIndexBuffer (3,ExtPDBuffer, INDICATOR_CALCULATIONS);
SetIndexBuffer (4,ExtNDBuffer, INDICATOR_CALCULATIONS);
// SetIndexBuffer (5,ExtTmpBuffer, INDICATOR_CALCULATIONS);
```

Теперь массив ExtTmpBuffer является просто динамическим массивом.

Откомпилируем код индикатора и присоединим индикатор к графику в терминале MetaTrader 5.

Терминал выдаст ошибку:

Time	Source	Message
2015.03.19 11:41:42.846	ADX (EURUSD,D1)	array out of range in 'ADX.mq5' (151,19)

Trade | Exposure | History | News ⁹⁹ | Mailbox | Market | Alerts | Signals | Code Base | **Experts** | Journal

Это произошло потому, что мы перед заполнением данного массива значениями не указали его размера и не зарезервировали под него память.

Так что его размер был равен нулю, когда мы попытались в него что-то записать.

Статическим мы этот массив сделать тоже не можем, т.е. объявить его сразу с указанием размера, так как значения такого промежуточного массива рассчитываются в функции обратного вызова OnCalculate на основе загруженной в функцию OnCalculate истории цен, а именно массивов open [], high [], low [], close [].

Но точный размер массивов open [], high [], low [], close [] неизвестен, он обозначается лишь переменной rates_total.

Хорошо, но мы можем в функции OnCalculate применить функцию ArrayResize, чтобы установить размер массива:

```
ArrayResize (ExtTmpBuffer, rates_total);
```

Теперь после компиляции индикатор заработает как надо.

Но дело в том, что в функции OnCalculate мы сначала рассчитываем индикатор для всей ценовой истории, т.е. для rates_total значений, а затем при поступлении нового тика по символу индикатора, и соответственно вызове функции OnCalculate, мы рассчитываем значение индикатора для этого нового тика по символу и записываем новое значение индикатора в его массив буфера.

Чтобы это реализовать с промежуточным массивом, нужно внимательно следить за его размером и записывать новое значение в конец массива.

Вместо всего этого, проще всего привязать промежуточный массив к буферу индикатора с помощью функции SetIndexBuffer и таким образом решить все эти проблемы.

Аналогичная ситуация возникает когда значения таких промежуточных массивов заполняются с помощью функции `CopyBuffer`, которая распределяет размер принимающего массива под размер копируемых данных.

Если копируется вся ценовая история, то проблем нет и в этом случае использовать `INDICATOR_CALCULATIONS` необязательно.

Если же мы хотим скопировать только одно новое поступившее значение, функция `CopyBuffer` определит размер принимающего массива как 1, и нужно будет использовать этот принимающий массив как еще один массив-посредник, из которого уже записывать значение в промежуточный массив индикатора. И в этом случае просто функцией `ArrayResize` для принимающего массива проблему не решить.

Теперь что нам делать, если мы хотим раскрашивать наши диаграммы индикатора в разные цвета в зависимости от цены?

Во-первых, мы должны указать, что наша графическая форма нашего графического построения является цветной, например:

```
#property indicator_type1 DRAW_COLOR_LINE
```

В идентификатор геометрической формы добавляется слово `COLOR`.

Далее значение свойства `#property indicator_buffers` увеличивается на единицу и объявляется еще один массив для хранения цвета.

Функцией `SetIndexBuffer` объявленный дополнительный массив сопоставляется с буфером цвета индикатора, например:

```
SetIndexBuffer (4,ExtColorsBuffer, INDICATOR_COLOR_INDEX);
```

В свойстве `#property indicator_color`, раскрашиваемого графического построения, указывается несколько цветов, например:

```
#property indicator_color1 Green, Red
```

И, наконец, каждому элементу массива, представляющего буфер цвета индикатора, присваивается номер цвета, определенный в свойстве `#property indicator_color`.

В данном случае, это 0.0 и 1.0.

Теперь при отрисовке диаграммы индикатора, из буфера берется значение диаграммы, по позиции значения оно сопоставляется со значением буфера цвета, и элемент диаграммы становится цветным.

Вместо свойства `#property indicator_color`, цвета графического построения можно задать программным способом:

```
//Задаем количество индексов цветов для графического построения
```

```
PlotIndexSetInteger (0,PLOT_COLOR_INDEXES,2);
```

```
//Задаем цвет для каждого индекса
```

```
PlotIndexSetInteger (0,PLOT_LINE_COLOR,0,Blue); //Нулевой индекс цвета – синий цвет
```

```
PlotIndexSetInteger (0,PLOT_LINE_COLOR,1,Orange); //Первый индекс цвета – оранжевый цвет
```

Где первый параметр – индекс графического построения, соответственно первое графическое построение имеет индекс 0.

Это идентично объявлению:

```
#property indicator_color1 Blue, Orange
```

Двинемся дальше по свойствам индикатора.

Толщина линии диаграммы индикатора задается свойством `indicator_widthN`, где N – номер графического построения, например:

```
#property indicator_width1 1
```

Также можно задать стиль линии диаграммы индикатора – сплошная линия, прерывистая, пунктирная, штрих-пунктирная, штрих – с помощью свойства `indicator_styleN`, где N – номер графического построения, например:

```
#property indicator_style1 STYLE_SOLID
```

И, наконец, свойство `indicator_labelN` указывает метки диаграмм индикатора в DataWindow или Окно данных, например:

```
#property indicator_label1 «ADX»
```

```
#property indicator_label2 "+DI»
```

```
#property indicator_label3 "-DI»
```

Другие свойства можно посмотреть в справочнике.

Правда можно отметить еще одну группу свойств, которая позволяет нарисовать горизонтальный уровень индикатора в отдельном окне, например:

```
#property indicator_level1 0.0
```

```
#property indicator_levelcolor Red
```

```
#property indicator_levelstyle STYLE_SOLID
```

```
#property indicator_levelwidth 2
```

В редакторе MQL5, в окне Navigator (Навигатор), в разделе Indicators-> Examples откроем исходный код индикатора ADX.

Блок объявления свойств индикатора выглядит следующим образом:

```
#property copyright «2009, MetaQuotes Software Corp.»
```

```
#property link "http://www.mql5.com"
```

```
#property description «Average Directional Movement Index»
```

```
#property indicator_separate_window
```

```
#property indicator_buffers 6
```

```
#property indicator_plots 3
```

```
#property indicator_type1 DRAW_LINE
```

```
#property indicator_color1 LightSeaGreen
```

```
#property indicator_style1 STYLE_SOLID
```

```
#property indicator_width1 1
```

```
#property indicator_type2 DRAW_LINE
```

```
#property indicator_color2 YellowGreen
```

```
#property indicator_style2 STYLE_DOT
```

```
#property indicator_width2 1
```

```
#property indicator_type3 DRAW_LINE
```

```
#property indicator_color3 Wheat
```

```
#property indicator_style3 STYLE_DOT
```

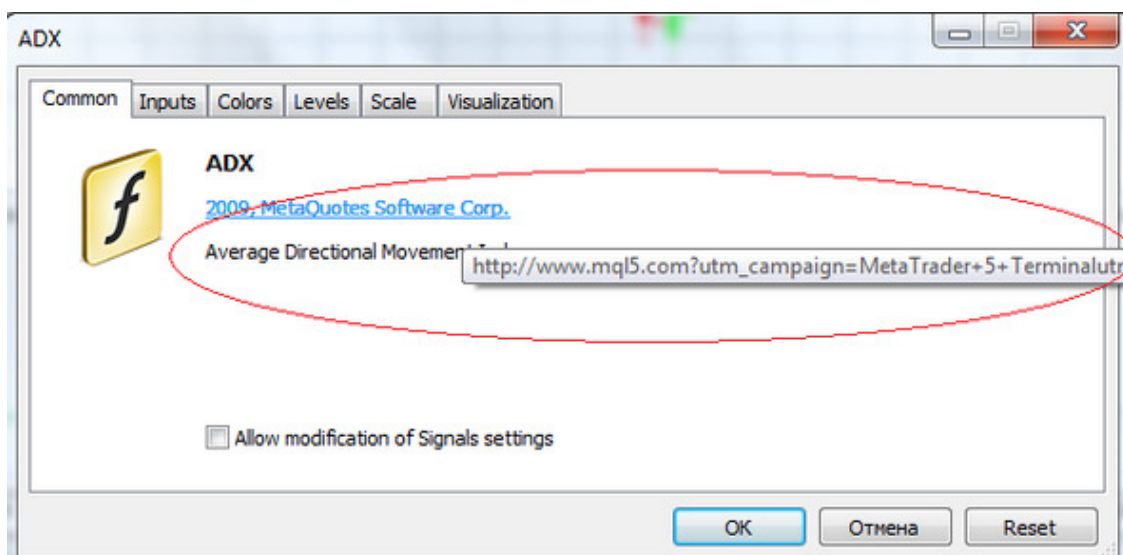
```
#property indicator_width3 1
```

```
#property indicator_label1 «ADX»
```

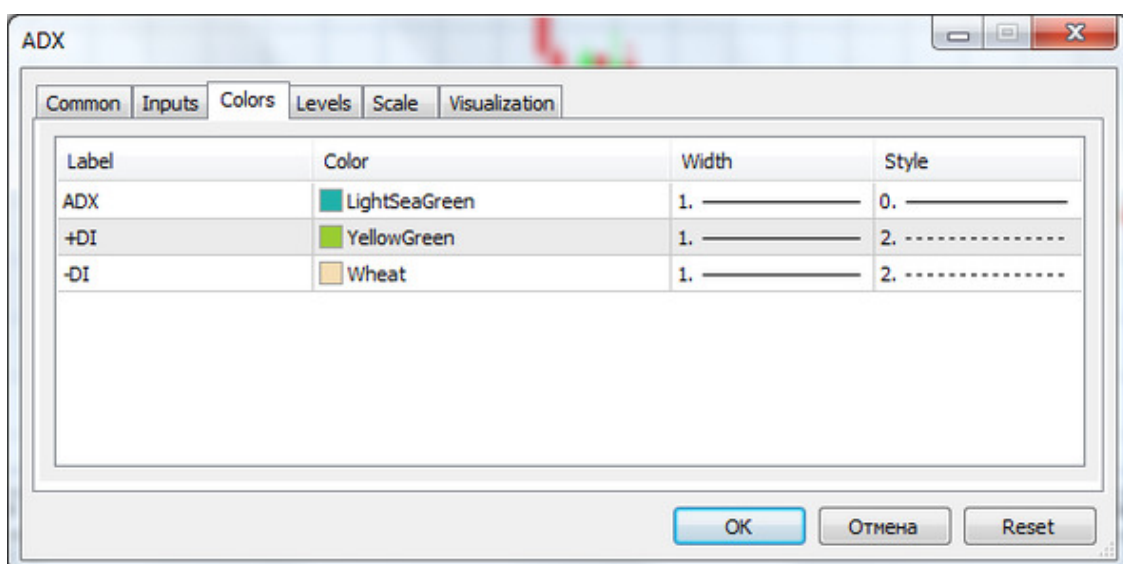
```
#property indicator_label2 "+DI»
```

```
#property indicator_label3 "-DI»
```

Если мы в MetaTrader 5 попытаемся присоединить данный индикатор к графику, во-первых, откроется диалоговое окно индикатора, которое во вкладке Common отобразит значения свойств `copyright`, `link` и `description`:



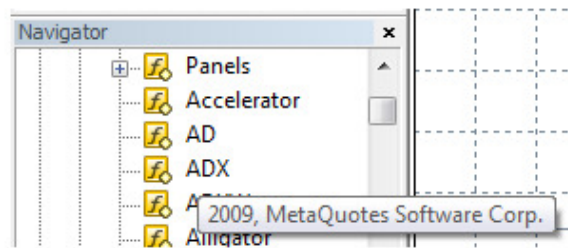
а во вкладке Colors отобразит значения свойств `indicator_label`, `indicator_color`, `indicator_width`, `indicator_style`:



Само же название индикатора определяется именем файла индикатора.

К слову сказать, диалоговое окно индикатора можно открыть и после присоединения индикатора к графику, с помощью контекстного меню, щелкнув правой кнопкой мышки на индикаторе и выбрав свойства индикатора.

При наведении курсора на название индикатора в окне Navigator терминала всплывает подсказка, отображающая свойство `copyright`.



После присоединения индикатора свойство:

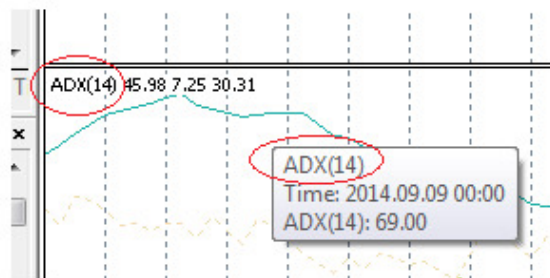
```
#property indicator_label1 «ADX»
```

работать не будет, так как в функции OnInit () с помощью вызова функции:

```
string short_name=«ADX (»+string (ExtADXPeriod) +»)»;
```

```
IndicatorSetString (INDICATOR_SHORTNAME, short_name);
```

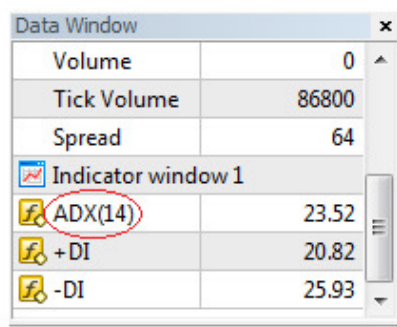
изменена подпись индикатора:



А вызовом функции:

```
PlotIndexSetString (0,PLOT_LABEL, short_name);
```

изменена метка индикатора в окне Data Window:



Data Window	
Volume	0
Tick Volume	86800
Spread	64
Indicator window 1	
ADX(14)	23.52
+DI	20.82
-DI	25.93

Значения же свойств:

```
#property indicator_label2 "+DI»
```

```
#property indicator_label3 "-DI»
```

отображаются, как и было определено, во всплывающих подсказках к диаграммам индикатора и отображаются в окне Data Window.

В коде индикатора ADX объявленное количество буферов индикатора больше, чем количество графических построений:

```
#property indicator_buffers 6
```

```
#property indicator_plots 3
```

Сделано это для того, чтобы использовать три буфера индикатора для промежуточных расчетов:

```
SetIndexBuffer (0,ExtADXBuffer);
```

```
SetIndexBuffer (1,ExtPDIBuffer);
```

```
SetIndexBuffer (2,ExtNDIBuffer);
```

```
SetIndexBuffer (3,ExtPDBuffer, INDICATOR_CALCULATIONS);
```

```
SetIndexBuffer (4,ExtNDBuffer, INDICATOR_CALCULATIONS);
```

```
SetIndexBuffer (5,ExtTmpBuffer, INDICATOR_CALCULATIONS);
```

В функции OnCalculate индикатора, значения массивов ExtPDBuffer, ExtNDBuffer, ExtTmpBuffer рассчитываются на основе загруженной ценовой истории, а затем уже на их основе рассчитываются значения массивов ExtADXBuffer, ExtPDIBuffer, ExtNDIBuffer, которые используются для отрисовки диаграмм индикатора.

Как уже было сказано, буфера индикатора для промежуточных вычислений здесь объявляются, так как заранее неизвестен размер загружаемой ценовой истории.

В описании индикатора ADX сказано, что:

Сигнал на покупку формируется тогда, когда +DI поднимается выше – DI и при этом сам ADX растет.

В момент, когда +DI расположен выше – DI, но сам ADX начинает снижаться, индикатор подает сигнал о том, что рынок «перегрет» и пришло время фиксировать прибыль.

Сигнал на продажу формируется тогда, когда +DI опускается ниже – DI и при этом ADX растет.

В момент, когда +DI расположен ниже – DI, но сам ADX начинает снижаться, индикатор подает сигнал о том, что рынок «перегрет» и пришло время фиксировать прибыль.

Давайте, модифицируем код индикатора ADX таким образом, чтобы раскрасить диаграмму ADX в четыре цвета, которые соответствуют описанным выше четырем торговым сигналам.

В качестве первого шага изменим свойство indicator_type1:

```
#property indicator_type1 DRAW_COLOR_LINE
```

Далее увеличим на единицу значение свойства `indicator_buffers`:

```
#property indicator_buffers 7
```

Объявим массив для буфера цвета:

```
double ExtColorsBuffer [];
```

В функции `OnInit ()` свяжем объявленный массив с буфером цвета:

```
SetIndexBuffer (0,ExtADXBuffer);
```

```
SetIndexBuffer (1,ExtColorsBuffer, INDICATOR_COLOR_INDEX);
```

```
SetIndexBuffer (2,ExtPDIBuffer);
```

```
SetIndexBuffer (3,ExtNDIBuffer);
```

```
SetIndexBuffer (4,ExtPDBuffer, INDICATOR_CALCULATIONS);
```

```
SetIndexBuffer (5,ExtNDBuffer, INDICATOR_CALCULATIONS);
```

```
SetIndexBuffer (6,ExtTmpBuffer, INDICATOR_CALCULATIONS);
```

Тут есть хитрость – **индекс буфера цвета должен следовать за индексом буфера значений индикатора**. Если, например, связать массив `ExtColorsBuffer` с буфером с индексом 6, тогда индикатор не будет корректно отрисовываться.

В свойство `indicator_color1` добавим цветов:

```
#property indicator_color1 LightSeaGreen, clrBlue, clrLightBlue, clrRed, clrLightPink
```

Увеличим толщину линии:

```
#property indicator_width1 2
```

В функции `OnCalculate` в конце перед закрывающей скобкой цикла `for` добавим код:

```
ExtColorsBuffer [i] =0;
```

```
if (ExtPDIBuffer [i]> ExtNDIBuffer [i] &&ExtADXBuffer [i]> ExtADXBuffer [i-1]) {  
ExtColorsBuffer [i] =1;
```

```
}
```

```
if (ExtPDIBuffer [i]> ExtNDIBuffer [i] &&ExtADXBuffer [i] <ExtADXBuffer [i-1]) {  
ExtColorsBuffer [i] =2;
```

```
}
```

```
if (ExtPDIBuffer [i] <ExtNDIBuffer [i] &&ExtADXBuffer [i]> ExtADXBuffer [i-1]) {  
ExtColorsBuffer [i] =3;
```

```
}
```

```
if (ExtPDIBuffer [i] <ExtNDIBuffer [i] &&ExtADXBuffer [i] <ExtADXBuffer [i-1]) {  
ExtColorsBuffer [i] =4;
```

```
}
```

Откомпилируем код и получим индикатор с визуальным отображением сигналов на покупку и продажу:



В редакторе MQL5 откроем другой индикатор из папки Examples – RSI.

Данный индикатор имеет два ключевых уровня, которые определяют области перекупленности и перепроданности.

В коде индикатора эти уровни определены как свойства:

```
#property indicator_level1 30
```

```
#property indicator_level2 70
```

Давайте улучшим отображение этих уровней, добавив им цвета и стиля.

Для этого добавим свойства:

```
#property indicator_levelcolor Red
```

```
#property indicator_levelstyle STYLE_SOLID
```

```
#property indicator_levelwidth 1
```

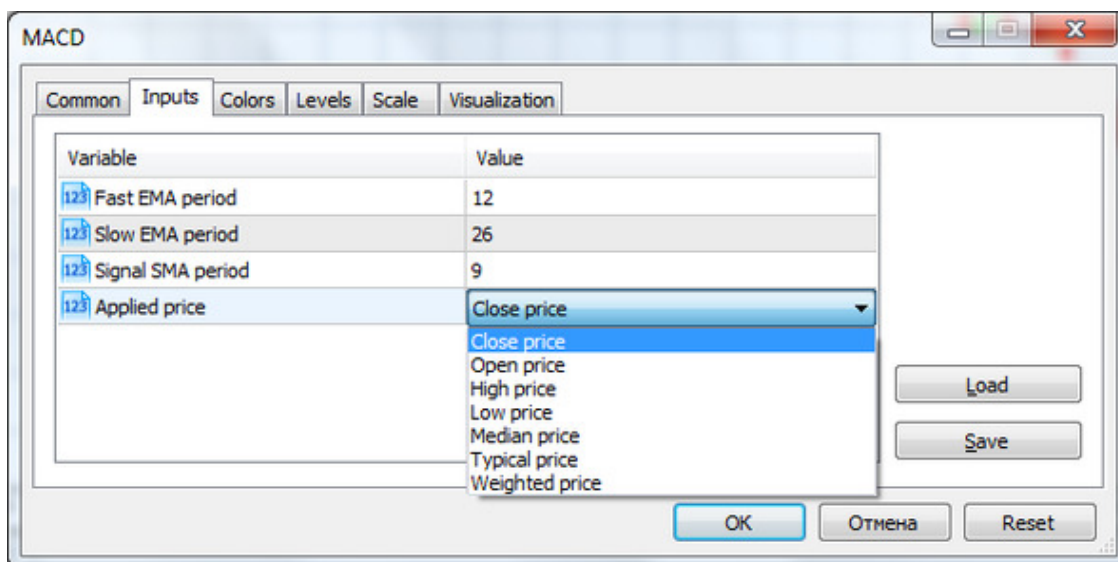
Теперь индикатор будет выглядеть следующим образом:



Параметры ввода и переменные индикатора

Параметры ввода это те параметры индикатора, которые отображаются пользователю перед присоединением индикатора к графику во вкладке Inputs диалогового окна.

Например, для индикатора MACD:



Тут пользователь может поменять параметры индикатора по умолчанию, и индикатор присоединится к графику с уже измененными параметрами.

Также пользователь может поменять параметры индикатора после присоединения индикатора к графику, щелкнув правой кнопкой мышки на индикаторе и выбрав свойства индикатора.

В коде индикатора такие параметры задаются Input переменными с модификатором input, который указывается перед типом данных. Как правило, Input переменные объявляются сразу после свойств индикатора.

Например, для индикатора MACD:

```
// - - input parameters
```

```
input int InpFastEMA=12; // Fast EMA period
```

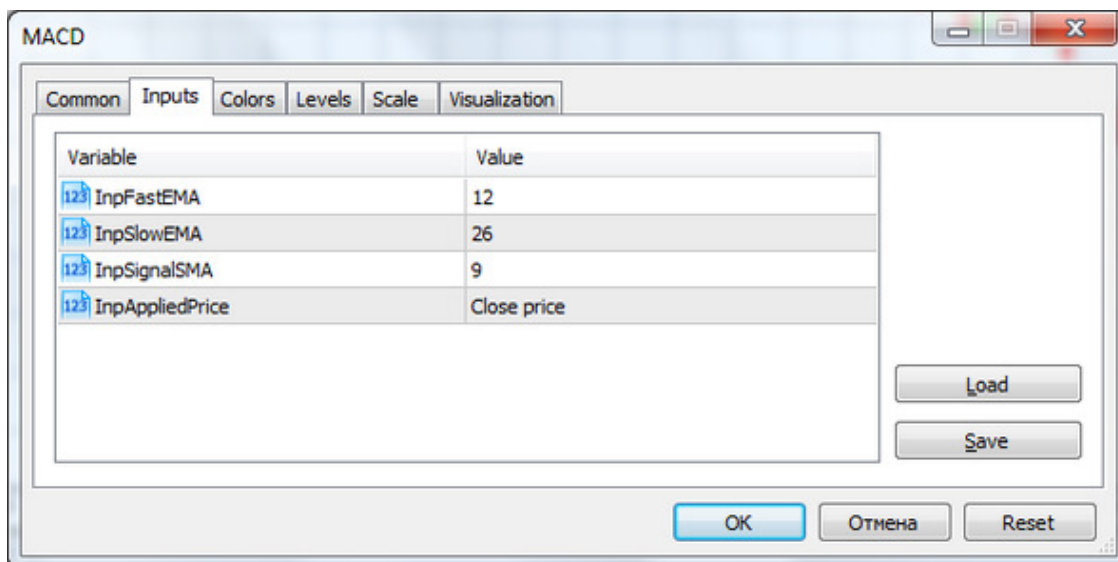
```
input int InpSlowEMA=26; // Slow EMA period
```

```
input int InpSignalSMA=9; // Signal SMA period
```

```
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE; // Applied price
```

Здесь надо отметить то, что в диалоговом окне присоединения индикатора к графику отображаются не имена переменных, а комментарии к ним.

Если убрать комментарии, входные параметры отобразятся следующим образом:



Здесь уже отображаются имена переменных.

Как вы сами, наверное, уже догадались, комментарии используются для отображения, чтобы облегчить пользователю понимание их предназначения.

Здесь также видно, что входными параметрами могут быть не только отдельные переменные, но и перечисления, которые отображаются в виде выпадающих списков.

Для индикатора MACD используется встроенное перечисление ENUM_APPLIED_PRICE, но можно также определить и свое перечисление.

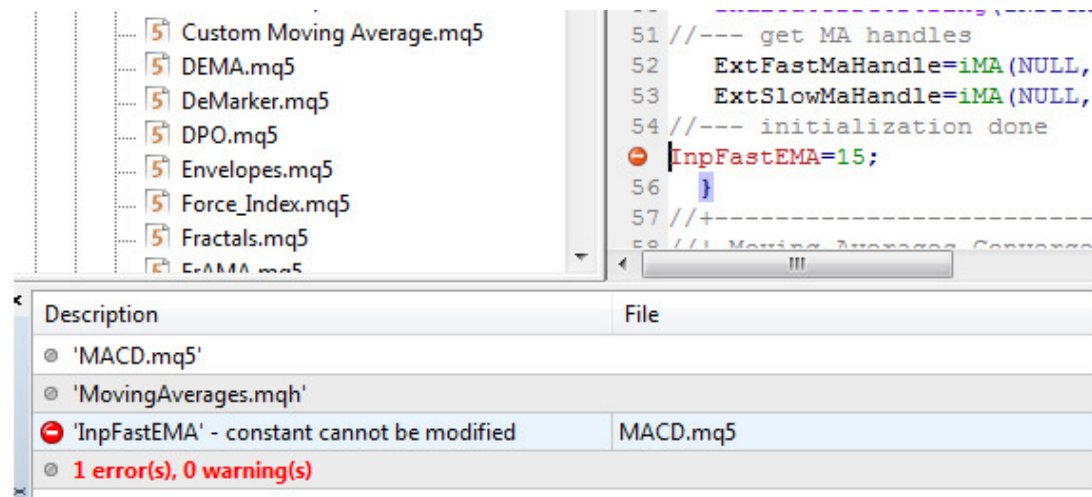
В справочнике приводится соответствующий пример:

```
#property script_show_inputs
// - - day of week
enum dayOfWeek
{
    S=0, // Sunday
    M=1, // Monday
    T=2, // Tuesday
    W=3, // Wednesday
    Th=4, // Thursday
    Fr=5, // Friday,
    St=6, // Saturday
};
// - - input parameters
input dayOfWeek swapday=W;
```

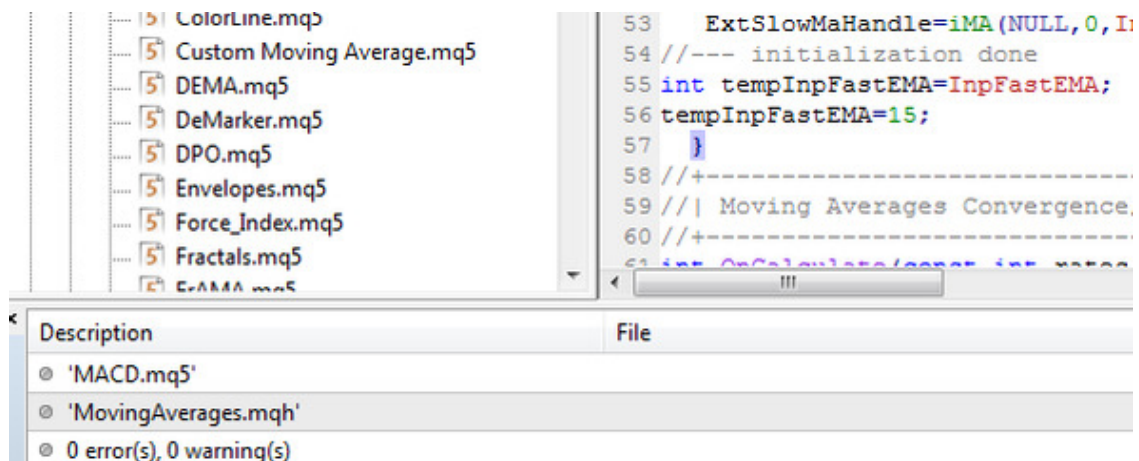
В этом примере команда #property script_show_inputs используется для скриптов, для индикаторов ее можно опустить.

Основное отличие Input переменных от других типов переменных состоит в том, что изменить их значение может только пользователь в диалоговом окне индикатора.

Если в коде индикатора попытаться изменить значение входного параметра, при компиляции возникнет ошибка:



Поэтому, если вы хотите при расчетах использовать измененное значение входного параметра, нужно использовать промежуточную переменную:



Помимо Input переменных MQL5-код использует локальные переменные, статические переменные, глобальные переменные и Extern переменные.

С локальными переменными в принципе все понятно, они объявляются в блоке кода, например, в цикле или функции, там же инициализируются, и, после выполнения блока кода, память, выделенная под локальные переменные в программном стеке, освобождается.

Тут особо надо отметить, что для локальных объектов, созданных с помощью оператора new, в конце блока кода нужно применить оператор delete для освобождения памяти.

Глобальные переменные, как правило, объявляются после свойств индикатора, входных параметров и массивов буферов индикатора, перед функциями.

Глобальные переменные видны в пределах всей программы, их значение может быть изменено в любом месте программы и память, выделяемая под глобальные переменные вне программного стека, освобождается при выгрузке программы.

Здесь видно, что Input переменные это те же глобальные переменные, за исключением опции – их значение не может быть изменено в любом месте программы.

Если глобальную или локальную переменную объявить со спецификатором const – это так же не позволит изменять значение этой переменной в процессе выполнения программы.

Статические переменные определяются модификатором static, который указывается перед типом данных.

Со статическими переменными все немного сложнее, но легче всего их понять, сравнивая статические переменные с локальными и глобальными переменными.

В принципе, статическая переменная, объявленная там же, где и глобальная переменная, ничем не отличается от глобальной переменной.

Хитрость начинается, если локальную переменную объявить с модификатором `static`.

В этом случае, после выполнения блока кода, память, выделенная под статическую переменную, не освобождается. И при следующем выполнении того же блока кода, предыдущее значение статической переменной можно использовать.

Хотя область видимости такой статической переменной ограничивается те же самым блоком кода, в котором она была объявлена.

Extern переменные это аналог статических глобальных переменных. Нельзя объявить локальную переменную с модификатором `extern`.

Отличие Extern переменных от статических глобальных переменных проще всего продемонстрировать на индикаторе MACD.

Индикатор MACD имеет включаемый файл `MovingAverages`:

```
#include <MovingAverages.mqh>
```

расположенный в папке `Include`.

Если в файле `MovingAverages` и файле `MACD` одновременно объявить Extern-переменную:

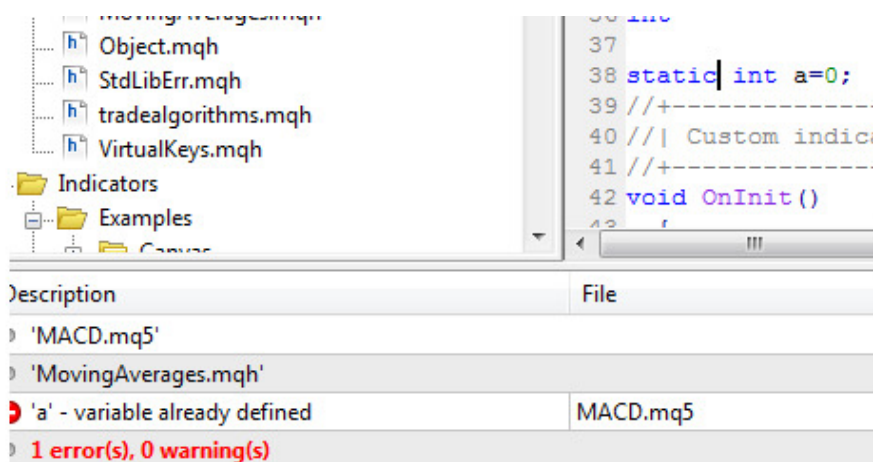
```
extern int a=0;
```

то при компиляции обоих файлов все пройдет удачно, и переменную можно будет использовать.

Если же в файле `MovingAverages` и файле `MACD` одновременно объявить статическую глобальную переменную:

```
static int a=0;
```

то при компиляции обоих файлов возникнет ошибка:



Помимо команды `#include` полезной является также директива `#define`, которая позволяет делать подстановку выражения вместо идентификатора, например:

```
#define PI 3.14
```

Хэндл индикатора

Начнем с цитаты:

HANDLE идентифицирует объект, которым Вы можете манипулировать. Джеффри РИХТЕР «Windows для профессионалов».

Переменные типа handle представляют собой указатель на некоторую системную структуру или индекс в некоторой системной таблице, которая содержит адрес структуры.

Таким образом, получив хэндл некоторого индикатора, мы можем использовать его данные для построения своего индикатора.

Хэндл индикатора представляет собой переменную типа int и объявляется, как правило, после объявления массивов буферов индикатора, вместе с глобальными переменными, например в индикаторе MACD:

```
// - indicator buffers
double ExtMacdBuffer [];
double ExtSignalBuffer [];
double ExtFastMaBuffer [];
double ExtSlowMaBuffer [];
// - MA handles
int ExtFastMaHandle;
int ExtSlowMaHandle;
```

Здесь хэндлы индикаторов – это указатели на индикатор скользящего среднего с разными периодами 12 и 26.

Объявив эти переменные, мы естественно реально ничего не получаем, так как объекта индикатора, данные которого мы хотим использовать, еще не существует.

Создать в глобальном кеше клиентского терминала копию соответствующего технического индикатора и получить ссылку на нее можно несколькими способами.

Если это стандартный индикатор, проще всего получить его хэндл можно с помощью стандартной функции для работы с техническими индикаторами.

Стандартная функция для индикатора скользящего среднего это:

```
int iMA (
    string symbol, // имя символа
    ENUM_TIMEFRAMES period, // период
    int ma_period, // период усреднения
    int ma_shift, // смещение индикатора по горизонтали
    ENUM_MA_METHOD ma_method, // тип сглаживания
    ENUM_APPLIED_PRICE applied_price // тип цены или handle
);
```

И в индикаторе MACD хэндлы индикатора скользящего среднего получают с помощью вызова функции iMA в функции OnInit ():

```
// - get MA handles
ExtFastMaHandle=iMA (NULL,0,InpFastEMA,0,MODE_EMA, InpAppliedPrice);
ExtSlowMaHandle=iMA (NULL,0,InpSlowEMA,0,MODE_EMA, InpAppliedPrice);
где используются свойства индикатора:
// - input parameters
input int InpFastEMA=12; // Fast EMA period
input int InpSlowEMA=26; // Slow EMA period
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE; // Applied price
```

Предположим, что мы хотим использовать не стандартный, а пользовательский индикатор.

В папке Indicators/Examples редактора MQL5 есть нужный нам индикатор – это файл Custom Moving Average.mq5.

Для вызова того индикатора воспользуемся функцией iCustom:

```
int iCustom (  
    string symbol, // имя символа  
    ENUM_TIMEFRAMES period, // период  
    string name // папка/имя_пользовательского индикатора  
    ...// список входных параметров индикатора  
);
```

В функции OnInit () индикатора MACD изменим код:

```
// ExtFastMaHandle=iMA (NULL,0,InpFastEMA,0,MODE_EMA, InpAppliedPrice);  
// ExtSlowMaHandle=iMA (NULL,0,InpSlowEMA,0,MODE_EMA, InpAppliedPrice);  
ExtFastMaHandle=iCustom (NULL,0,«Examples\\Custom Moving Average»,  
InpFastEMA,0,MODE_EMA, InpAppliedPrice);  
ExtSlowMaHandle=iCustom (NULL,0,«Examples\\Custom Moving Average»,  
InpSlowEMA,0,MODE_EMA, InpAppliedPrice);
```

После компиляции индикатора мы увидим, что его отображение никак не изменилось:



Еще один способ получить хэндл пользовательского индикатора, это использовать функцию IndicatorCreate:

```
int IndicatorCreate (  
    string symbol, // имя символа  
    ENUM_TIMEFRAMES period, // период  
    ENUM_INDICATOR indicator_type, // тип индикатора из перечисления  
    ENUM_INDICATOR
```

```
int parameters_cnt=0, // количество параметров
const MqlParam& parameters_array [] =NULL, // массив параметров
);
В функции OnInit () индикатора MACD изменим код:
MqlParam params [];
ArrayResize (params,5);
params [0].type =TYPE_STRING;
params[0].string_value=«Examples\\Custom Moving Average»;
// - - set ma_period
params [1].type =TYPE_INT;
params[1].integer_value=InpFastEMA;
// - - set ma_shift
params [2].type =TYPE_INT;
params[2].integer_value=0;
// - - set ma_method
params [3].type =TYPE_INT;
params[3].integer_value=MODE_EMA;
// - - set applied_price
params [4].type =TYPE_INT;
params[4].integer_value=InpAppliedPrice;
// - - initialization done
ExtFastMaHandle=IndicatorCreate (NULL, NULL, IND_CUSTOM,4,params);
params[1].integer_value=InpSlowEMA;
ExtSlowMaHandle=IndicatorCreate (NULL, NULL, IND_CUSTOM,4,params);
```

После компиляции индикатора мы опять увидим, что его отображение никак не изменилось.

После получения хэндла индикатора, если он используется в коде один раз, для экономии памяти неплохо использовать функцию IndicatorRelease:

```
bool IndicatorRelease (
int indicator_handle // handle индикатора
);
которая удаляет хэндл индикатора и освобождает расчетную часть индикатора.
Хорошо, хэндл индикатора мы получили. Как же теперь извлечь его данные?
Делается это в функции OnCalculate с помощью функции CopyBuffer:
int CopyBuffer (
int indicator_handle, // handle индикатора
int buffer_num, // номер буфера индикатора
int start_pos, // откуда начнем
int count, // сколько копируем
double buffer [] // массив, куда будут скопированы данные
);
```

При этом функция CopyBuffer () распределяет размер принимающего массива под размер копируемых данных.

Напомним, что это работает, если принимающий массив является просто динамическим массивом.

Если же принимающий массив связан с буфером индикатора, тогда клиентский терминал сам заботится о том, чтобы размер такого массива соответствовал количеству баров, доступных индикатору для расчета.

В индикаторе MACD именно такая ситуация. Промежуточные массивы ExtFastMaBuffer [] и ExtSlowMaBuffer [] привязаны к буферам индикатора:

```
SetIndexBuffer (2,ExtFastMaBuffer, INDICATOR_CALCULATIONS);  
SetIndexBuffer (3,ExtSlowMaBuffer, INDICATOR_CALCULATIONS);
```

И в эти массивы производится копирование буфера индикатора Moving Average на основе его хэндлов:

```
if (CopyBuffer (ExtFastMaHandle,0,0,to_copy, ExtFastMaBuffer) <=0)  
{  
    Print («Getting fast EMA is failed! Error», GetLastError ());  
    return (0);  
}  
if (CopyBuffer (ExtSlowMaHandle,0,0,to_copy, ExtSlowMaBuffer) <=0)  
{  
    Print («Getting slow SMA is failed! Error», GetLastError ());  
    return (0);  
}
```

Если убрать привязку массивов ExtFastMaBuffer [] и ExtSlowMaBuffer [] к буферам индикатора, тогда клиентский терминал выдаст ошибку:

Time	Source	Message
2015.03.19 19:14:51.657	MACD (EURUSD,D1)	array out of range in 'MACD.mq5' (119,38)

Trade | Exposure | History | News ⁹⁹ | Mailbox | Market | Alerts | Signals | Code Base | **Experts** | Journal

Происходит это потому, что при загрузке индикатора значение to_copy равно размеру ценовой истории, а дальше to_copy=1 и производится частичное копирование в массивы ExtFastMaBuffer [] и ExtSlowMaBuffer [], при этом их размеры становятся равны 1.

В этом случае применением функции ArrayResize проблему не решить, так как функция CopyBuffer все равно будет уменьшать размер массива до 1.

Можно конечно использовать еще один массив-посредник, в который копировать один элемент. И уже из этого массива-посредника производить копирование в промежуточный массив, но проще всего, конечно, просто привязать промежуточный массив к буферу индикатора.

Функции OnInit (), OnDeinit (), OnCalculate ()

Как уже говорилось, функции OnInit (), OnDeinit (), OnCalculate () вызываются клиентским терминалом при наступлении определенных событий.

Функция OnInit ()

Функция OnInit () вызывается сразу после загрузки индикатора и соответственно используется для его инициализации.

Инициализация индикатора включает в себя привязку массивов к буферам индикатора, инициализацию глобальных переменных, включая инициализацию хэндлеров используемых индикаторов, а также программную установку свойств индикатора.

Давайте разберем некоторые из этих пунктов более подробно.

Как уже было показано, привязка массивов к буферам индикатора осуществляется с помощью функции SetIndexBuffer:

```
bool SetIndexBuffer (  
    int index, // индекс буфера  
    double buffer [], // массив  
    ENUM_INDEXBUFFER_TYPE data_type // что будем хранить  
);
```

Где data_type может быть INDICATOR_DATA (данные индикатора для отрисовки, по умолчанию, можно не указывать), INDICATOR_COLOR_INDEX (цвет индикатора), INDICATOR_CALCULATIONS (буфер промежуточных расчетов индикатора).

После применения функции SetIndexBuffer к динамическому массиву, его размер автоматически поддерживается равным количеству баров, доступных индикатору для расчета.

Каждый индекс массива типа INDICATOR_COLOR_INDEX соответствует индексу массива типа INDICATOR_DATA, а значение индекса массива типа INDICATOR_COLOR_INDEX определяет цвет отображения индекса массива типа INDICATOR_DATA.

Значение индекса массива типа INDICATOR_COLOR_INDEX, при его установке, берется из свойства #property indicator_colorN как индекс цвета в строке.

Индекс буфера типа INDICATOR_COLOR_INDEX должен следовать за индексом буфера типа INDICATOR_DATA.

После привязки динамического массива к буферу индикатора можно поменять порядок доступа к массиву от конца к началу, т.е. значение массива с индексом 0 будет соответствовать последнему полученному значению индикатора. Сделать это можно с помощью функции ArraySetAsSeries:

```
bool ArraySetAsSeries (  
    const void& array [], // массив по ссылке  
    bool flag // true означает обратный порядок индексации  
);
```

При применении функции ArraySetAsSeries физическое хранение данных массива не меняется, в памяти массив, как и прежде, хранится в порядке от первого значения до последнего значения.

Функция ArraySetAsSeries меняет лишь программный доступ к элементам массива — от последнего элемента массива к первому элементу массива.

В функции OnInit () также может осуществляться проверка входных параметров на корректность, так как пользователь может ввести все, что угодно.

При этом значение входного параметра переназначается с помощью глобальной переменной, и далее в расчетах используется уже значение глобальной переменной.

Например, для индикатора ADX это выглядит так:

```
// - - check for input parameters  
if (InpPeriodADX >= 100 || InpPeriodADX <= 0)
```

```
{
    ExtADXPeriod=14;
    printf («Incorrect value for input variable Period_ADX=%d. Indicator will use value=%d for
calculations.», InpPeriodADX, ExtADXPeriod);
}
```

```
else ExtADXPeriod=InpPeriodADX;
```

здесь ExtADXPeriod – глобальная переменная, а InpPeriodADX – входной параметр.

При использовании хэндлов индикатора, необходимо указывать символ (финансовый инструмент) для которого индикатор будет создаваться.

При этом такой символ может определяться пользователем.

В функции OnInit () также полезно проверить этот входной параметр на корректность.

Пусть определен входной параметр:

```
input string symbol=" "; // символ
```

Объявим глобальную переменную:

```
string name=symbol;
```

В функции OnInit () произведем проверку:

```
// - - удалим пробелы слева и справа
```

```
StringTrimRight (name);
```

```
StringTrimLeft (name);
```

```
// - - если после этого длина строки name нулевая
```

```
if (StringLen (name) ==0)
```

```
{
```

```
// - - возьмем символ с графика, на котором запущен индикатор
```

```
name=_Symbol;
```

```
}
```

Программная установка свойств индикатора осуществляется с помощью функций IndicatorSetDouble, IndicatorSetInteger, IndicatorSetString, PlotIndexSetDouble, PlotIndexSetInteger, PlotIndexSetString.

Функция IndicatorSetDouble позволяет программным способом определять такие свойства индикатора как indicator_minimum, indicator_maximum и indicator_levelN, например:

```
IndicatorSetDouble (INDICATOR_LEVELVALUE, 0, 50)
```

является аналогом:

```
property indicator_level1 50
```

Функция IndicatorSetInteger позволяет программным способом определять такие свойства индикатора как indicator_height, indicator_levelcolor, indicator_levelwidth, indicator_levelstyle.

При этом для уровней необходимо определить их количество, используя функцию IndicatorSetInteger. Например, для индикатора RSI это выглядит следующим образом.

Свойства индикатора:

```
##property indicator_level1 30
```

```
##property indicator_level2 70
```

```
##property indicator_levelcolor Red
```

```
##property indicator_levelstyle STYLE_SOLID
```

```
##property indicator_levelwidth 1
```

Заменяем на код:

```
IndicatorSetInteger (INDICATOR_LEVELS,2);
```

```
IndicatorSetDouble (INDICATOR_LEVELVALUE,0,30);
```

```
IndicatorSetDouble (INDICATOR_LEVELVALUE,1,70);
```

```
IndicatorSetInteger (INDICATOR_LEVELCOLOR,0,0xff0);
```

```
IndicatorSetInteger (INDICATOR_LEVELCOLOR,1,0xff0);  
IndicatorSetInteger (INDICATOR_LEVELSTYLE,0,STYLE_SOLID);  
IndicatorSetInteger (INDICATOR_LEVELSTYLE,1,STYLE_SOLID);  
IndicatorSetInteger (INDICATOR_LEVELWIDTH,0,1);  
IndicatorSetInteger (INDICATOR_LEVELWIDTH,1,1);
```

Функция `IndicatorSetInteger` также позволяет определить точность индикатора, например:

```
IndicatorSetInteger (INDICATOR_DIGITS,2);
```

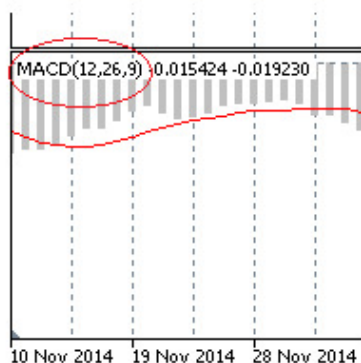
В результате будут отображаться только два знака после запятой значения индикатора.

Для функции `IndicatorSetString` нет соответствующих ей свойств индикатора.

С помощью функции `IndicatorSetString` можно определить короткое наименование индикатора, например для индикатора MACD:

```
IndicatorSetString (INDICATOR_SHORTNAME,«MACD («»+string (InpFastEMA)  
+»,»+string (InpSlowEMA) +»,»+string (InpSignalSMA) +»)»);
```

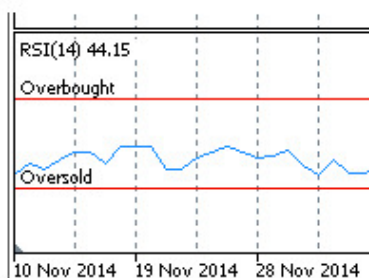
Соответственно имя индикатора будет отображаться в окне индикатора как:



Кроме того, функция `IndicatorSetString` позволяет установить подписи к уровням индикатора, например для индикатора RSI:

```
IndicatorSetString (INDICATOR_LEVELTEXT, 0,«Oversold»);
```

```
IndicatorSetString (INDICATOR_LEVELTEXT, 1,«Overbought»)
```



С помощью функции `PlotIndexSetDouble` определяют, какое значение буфера индикатора является пустым и не участвует в отрисовке диаграммы индикатора.

Диаграмма индикатора рисуется от одного непустого значения до другого непустого значения индикаторного буфера, пустые значения пропускаются. Чтобы указать, какое значение следует считать «пустым», необходимо определить это значение в свойстве `PLOT_EMPTY_VALUE`. Например, если индикатор должен рисоваться по ненулевым значениям, то нужно задать нулевое значение в качестве пустого значения буфера индикатора:

```
PlotIndexSetDouble (индекс_построения, PLOT_EMPTY_VALUE,0);
```

Функция `PlotIndexSetInteger` позволяет программным способом, динамически, задавать такие свойства диаграммы индикатора, как код стрелки для стиля `DRAW_ARROW`, смещение стрелок по вертикали для стиля `DRAW_ARROW`, количество начальных баров без отрисовки и значений в `DataWindow`, тип графического построения, признак отображения значений построения в окне `DataWindow`, сдвиг графического построения индикатора по оси времени в барах, стиль линии отрисовки, толщина линии отрисовки, количество цветов, индекс буфера, содержащего цвет отрисовки.

Давайте разберем каждое из этих свойств по порядку на примере индикатора `Custom Moving Average`.

Изменим свойство `indicator_type1` индикатора `Custom Moving Average`:

```
#property indicator_type1 DRAW_ARROW
```

В функции `OnInit ()` добавим вызов функции `PlotIndexSetInteger`, определяя различный код стрелки для стиля `DRAW_ARROW`:

```
PlotIndexSetInteger (0,PLOT_ARROW,2);
```



```
PlotIndexSetInteger (0,PLOT_ARROW,3);
```



```
PlotIndexSetInteger (0,PLOT_ARROW,4);
```



```
PlotIndexSetInteger (0,PLOT_ARROW,5);
```



```
PlotIndexSetInteger (0,PLOT_ARROW,6);
```



```
PlotIndexSetInteger (0,PLOT_ARROW,7);
```



```
PlotIndexSetInteger (0,PLOT_ARROW,8);
```



```
PlotIndexSetInteger (0,PLOT_ARROW,11);
```



```
PlotIndexSetInteger (0,PLOT_ARROW,12);
```



```
PlotIndexSetInteger (0,PLOT_ARROW,14);
```



```
PlotIndexSetInteger (0,PLOT_ARROW,15);
```



И так далее. Я думаю, этого будет достаточно для демонстрации этой опции.

В функции OnInit () добавим вызов функции PlotIndexSetInteger, определяя смещение стрелок по вертикали для стиля DRAW_ARROW:

```
PlotIndexSetInteger (0,PLOT_ARROW_SHIFT,0);
```



```
PlotIndexSetInteger (0,PLOT_ARROW_SHIFT,100)
```



В результате диаграмма индикатора сдвинулась вниз.

В индикаторе Custom Moving Average для определения количества начальных баров без отрисовки и значений в DataWindow используется вызов функции PlotIndexSetInteger:

```
PlotIndexSetInteger (0,PLOT_DRAW_BEGIN, InpMAPeriod);
```

где InpMAPeriod – период скользящей средней.

Идентификатор свойства PLOT_DRAW_TYPE функции PlotIndexSetInteger позволяет программным способом задать свойство индикатора indicator_typeN, например:

```
PlotIndexSetInteger (0, PLOT_DRAW_TYPE, DRAW_ARROW);
```

Причем, если одновременно задано свойство indicator_typeN и сделан вызов функции PlotIndexSetInteger с идентификатором PLOT_DRAW_TYPE – действовать будет тип диаграммы, заданный функцией PlotIndexSetInteger.

Убрать отображение текущих значений диаграммы индикатора в окне DataWindow при наведении курсора мышки можно с помощью вызова функции PlotIndexSetInteger с идентификатором PLOT_SHOW_DATA:

```
PlotIndexSetInteger (0, PLOT_SHOW_DATA, false);
```

В индикаторе Custom Moving Average для определения сдвига графического построения индикатора по оси времени в барах используется вызов функции PlotIndexSetInteger:

```
PlotIndexSetInteger (0,PLOT_SHIFT, InpMAShift);
```

При InpMAShift=0:



При InpMAShift=10:



Такой сдвиг делается для имитации предсказательности индикатора.

Идентификатор свойства PLOT_LINE_STYLE функции PlotIndexSetInteger позволяет программным способом задать свойство индикатора indicator_styleN, стиль линии отрисовки, например:

```
PlotIndexSetInteger (0, PLOT_LINE_STYLE, STYLE_DASHDOT);
```



Идентификатор свойства PLOT_LINE_WIDTH функции PlotIndexSetInteger позволяет программным способом задать свойство индикатора indicator_widthN, толщину линии отрисовки, например:

```
PlotIndexSetInteger (0, PLOT_LINE_WIDTH, 2);
```

Программным способом задать свойство индикатора indicator_colorN позволяет вызов функции PlotIndexSetInteger с идентификаторами PLOT_COLOR_INDEXES и PLOT_LINE_COLOR, например:

```
#property indicator_color1 clrGreen, clrRed
```

Или

```
PlotIndexSetInteger (0,PLOT_COLOR_INDEXES,2);
```

```
PlotIndexSetInteger (0,PLOT_LINE_COLOR,0,clrGreen);
```

```
PlotIndexSetInteger (0,PLOT_LINE_COLOR,1,clrRed);
```

Функция PlotIndexSetString позволяет программным способом задать свойство индикатора indicator_labelN. Например, для индикатора MACD это будет выглядеть следующим образом:

```
#property indicator_label1 «MACD»
```

```
#property indicator_label2 «Signal»
```

Или

```
PlotIndexSetString (0, PLOT_LABEL, «MACD»);
```

```
PlotIndexSetString (1, PLOT_LABEL, «Signal»);
```

Рассмотренные выше функции программной установки свойств индикатора можно конечно вызывать и в функции обратного вызова OnCalculate, но глубокого смысла в этом нет, так как они не могут быть применены только к части диаграммы индикатора – они применяются сразу ко всей диаграмме индикатора. Поэтому для экономии ресурсов лучше всего вызывать эти функции в функции обратного вызова OnInit ().

Функция OnDeinit ()

Прочитируем справочник:

Событие Deinit генерируется для экспертов и индикаторов в следующих случаях:

- перед переинициализацией в связи со сменой символа или периода графика, к которому прикреплен mql5-программа;
- перед переинициализацией в связи со сменой входных параметров;
- перед выгрузкой mql5-программы.

Так как функция OnDeinit () вызывается при деинициализации, то ее основное предназначение, это освобождение занимаемых ресурсов.

Под освобождением занимаемых ресурсов для индикатора подразумевается очищение графика символа от дополнительных графических объектов.

То есть помимо диаграммы индикатора, мы можем присоединять к графику символа различные объекты – линии, графические фигуры треугольник, прямоугольник и эллипс, знаки, подписи и др. Об этом мы поговорим позже.

Соответственно при деинициализации индикатора было бы неплохо все это убрать с графика символа.

Первым делом здесь используется функция Comment, которая выводит комментарий, определенный пользователем, в левый верхний угол графика:

```
void Comment (  
    argument, // первое значение  
    ...// последующие значения  
);
```

Для очистки от комментариев используются пустые комментарии:

```
Comment («»);
```

Далее используется функция ObjectDelete, которая удаляет объект с указанным именем с указанного графика:

```
bool ObjectDelete (  
    long chart_id, // chart identifier  
    string name // object name  
);
```

Позже мы продемонстрируем применение этих функций.

Функция OnCalculate ()

Функция OnCalculate () вызывается клиентским терминалом при поступлении нового тика по символу, для которого рассчитывается индикатор.

Хотя функция OnCalculate () имеет два вида – для индикатора, который может быть рассчитан на основе только одной из ценовых таймсерий:

```
int OnCalculate (const int rates_total, // размер массива price []
const int prev_calculated, // обработано баров на предыдущем вызове
const int begin, // откуда начинаются значимые данные
const double& price [] // массив для расчета
);
```

и для индикатора, который рассчитывается с использованием нескольких ценовых таймсерий:

```
int OnCalculate (const int rates_total, // размер входных таймсерий
const int prev_calculated, // обработано баров на предыдущем вызове
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
);
```

Здесь мы будем пользоваться полной версией функции OnCalculate () как наиболее гибкой и предоставляющей наибольшие возможности.

Единственное, что мы должны отметить об усеченной функции OnCalculate (), это то, что она имеет опцию использования в качестве массива price [] рассчитанного буфера другого индикатора.

Продемонстрируем это на примере индикатора MACD и индикатора Custom Moving Average, который использует как раз усеченную функцию OnCalculate ().

Присоединим сначала индикатор MACD к графику символа, а затем перетащим индикатор МА в окно индикатора MACD:



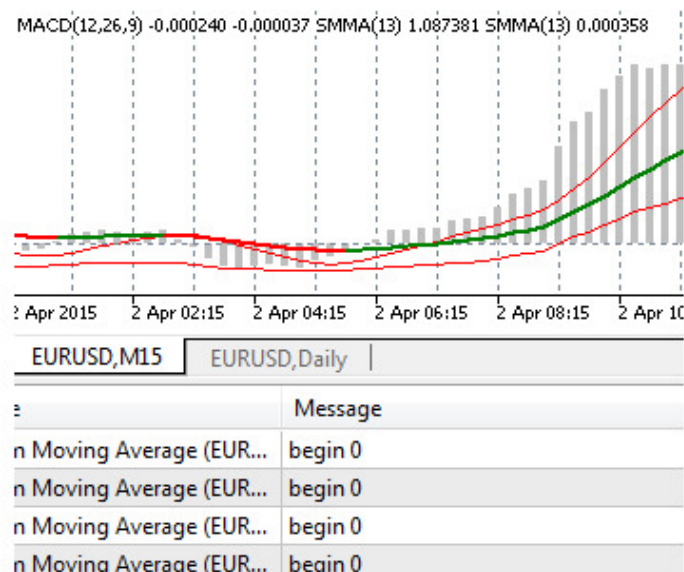
Затем опять перетащим индикатор Custom Moving Average в окно индикатора MACD, при этом снова откроется окно параметров индикатора Custom Moving Average:



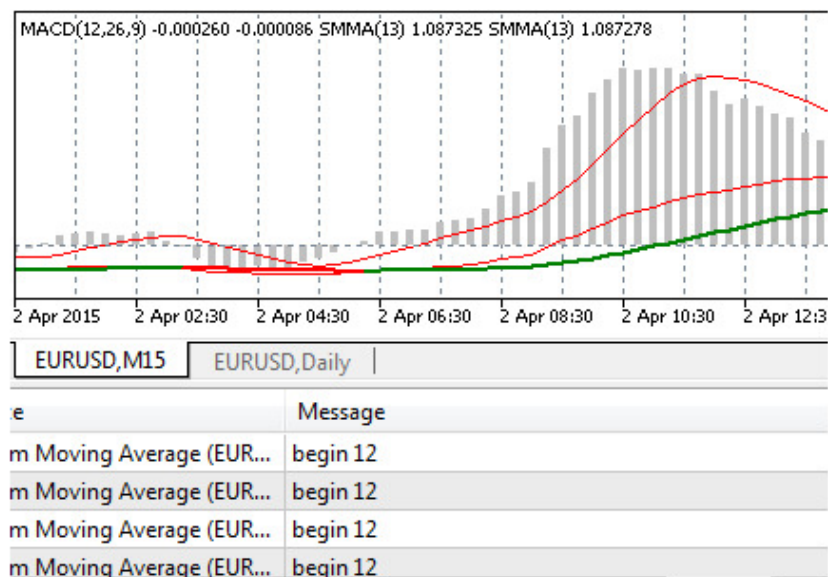
В списке выбора, что использовать в качестве массива price [], будут пункты First Indicator's Data и Previous Indicator's Data.

Здесь пункт First Indicator's Data означает, что в качестве массива price [] будет использоваться массив ExtMacdBuffer буфера индикатора MACD, а пункт Previous Indicator's Data означает, что в качестве массива price [] будет использоваться массив ExtLineBuffer буфера индикатора MA.

Если в функцию OnCalculate индикатора Custom Moving Average добавить:
 Print («begin», begin);
 То при выборе First Indicator's Data будет выводиться:



А при выборе Previous Indicator's Data будет выводиться:



В первом случае, begin=0, так как для буфера ExtMacdBuffer индикатора MACD функция PlotIndexSetInteger с параметром PLOT_DRAW_BEGIN не вызывается. А во втором случае, begin=12, так как для буфера ExtLineBuffer индикатора МА вызывается функция PlotIndexSetInteger:

PlotIndexSetInteger (0,PLOT_DRAW_BEGIN, InpMAPeriod-1+begin);

Тут говорится о том, что массив буфера ExtLineBuffer индикатора МА заполняется, начиная с InpMAPeriod-1 бара, соответственно значение переменной begin функции OnCalculate индикатора Custom Moving Average будет также равно InpMAPeriod-1.

Вернемся к полной версии функции OnCalculate ().

Как правило, код функции OnCalculate () проектируется таким образом, чтобы при загрузке индикатора и первом вызове функции OnCalculate (), буфера индикатора были рассчитаны на основе всей загруженной ценовой истории, а при последующем поступлении нового тика и вызове функции OnCalculate (), рассчитывалось бы только одно новое значение, которое добавляется в конец массива буфера индикатора.

Но в начале кода функции OnCalculate () нужно конечно проверить, достаточный ли размер ценовой истории был загружен при загрузке индикатора.

Для этого проверяется значение переменной rates_total – размер входных таймсерий.

Как правило, в качестве порогового значения для rates_total принимается значение периода индикатора, например для индикатора ADX:

```
// - - checking for bars count
if (rates_total < ExtADXPeriod)
return (0);
```

Если же в расчете буфера индикатора участвует хэндл другого индикатора, тогда проверяется количество рассчитанных данных для запрашиваемого индикатора:

```
// - - узнаем количество рассчитанных значений в индикаторе
int calculated=BarsCalculated (handle);
if (calculated <=0)
{
PrintFormat («BarsCalculated () вернул %d, код ошибки %d», calculated, GetLastError ());
return (0);
}
```

После проверки первоначальной загруженной истории для расчетов, вычисляется размер данных, которые необходимо рассчитать в этом вызове функции OnCalculate ().

В качестве примера, разберем блок кода, который приводится в справочнике, в разделе Технические индикаторы:

```
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 [])
{
// - - количество копируемых значений из индикатора
int values_to_copy;
// - - узнаем количество рассчитанных значений в индикаторе
int calculated=BarsCalculated (handle);
if (calculated <=0)
{
PrintFormat («BarsCalculated () вернул %d, код ошибки %d», calculated, GetLastError ());
return (0);
}
```

// - - если это первый запуск вычислений нашего индикатора или изменилось количество значений в индикаторе

```
// - или если необходимо рассчитать индикатор для двух или более баров (значит что-
то изменилось в истории)
if (prev_calculated==0 || calculated!=bars_calculated || rates_total> prev_calculated+1)
{
    // - если массив больше, чем значений в индикаторе на паре symbol/period, то копи-
руем не все
    // - в противном случае копировать будем меньше, чем размер индикаторных буферов
    if (calculated> rates_total) values_to_copy=rates_total;
    else values_to_copy=calculated;
}
else
{
    // - значит наш индикатор рассчитывается не в первый раз и с момента последнего
вызова OnCalculate ()
    // - для расчета добавилось не более одного бара
    values_to_copy= (rates_total-prev_calculated) +1;
}

// - заппомним количество значений в индикаторе
bars_calculated=calculated;
// - вернем значение prev_calculated для следующего вызова
return (rates_total);
}
```

Здесь переменная `values_to_copy` – количество рассчитываемых значений в вызове функции `OnCalculate ()`.

Переменная `prev_calculated` – сколько было обработано баров функцией `OnCalculate ()` при предыдущем вызове.

Таким образом, при загрузке индикатора `prev_calculated=0`, а при каждом следующем поступлении нового тика `prev_calculated= rates_total`.

Переменная `prev_calculated` также обнуляется терминалом, если вдруг изменилось значение переменной `rates_total`.

Переменная `bars_calculated` – предыдущее количество рассчитанных данных для запрашиваемого индикатора, на основе которого рассчитывается данный индикатор.

Таким образом, первая проверка здесь:

`prev_calculated==0` – индикатор только что загрузился или изменилась ценовая история.

`calculated!=bars_calculated` – изменилось количество рассчитанных данных для запрашиваемого индикатора.

`rates_total> prev_calculated+1` – необходимо рассчитать индикатор для двух или более баров (значит, что-то изменилось в истории).

Последнее условие вступает в противоречие с утверждением справочника:

Если с момента последнего вызова функции `OnCalculate ()` ценовые данные были изменены (подкачана более глубокая история или были заполнены пропуски истории), то значение входного параметра `prev_calculated` будет установлено в нулевое значение самим терминалом.

Если изменилась история, тогда сработает проверка `prev_calculated==0` и проверка последнего условия будет излишней.

Теперь, если срабатывает первое или второе условие, тогда количество рассчитываемых значений – это размер входных таймсерий или количество рассчитанных данных для

запрашиваемого индикатора, на основе которого рассчитывается данный индикатор, что из них меньше.

Если же первое или второе условие не срабатывают, тогда количество рассчитываемых значений:

```
values_to_copy= (rates_total-prev_calculated) +1;
```

Опять же, тут есть излишний код, так как, судя по справочнику, переменная prev_calculated может принимать значение либо 0, либо rates_total.

Поэтому, values_to_copy=1.

Таким образом, при поступлении нового тика, будет рассчитываться только одно значение индикатора для этого нового тика.

Рассмотрим другую реализацию вычисления размера данных, которые необходимо рассчитать в вызове функции OnCalculate ().

Для индикатора MACD это реализовано следующим образом:

```
// - - 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;
```

```
if (prev_calculated> 0) to_copy++;
```

```
}
```

Опять же, судя по справочнику, здесь будет работать только код:

```
to_copy=rates_total-prev_calculated;
```

```
if (prev_calculated> 0) to_copy++;
```

Т.е. при загрузке индикатора to_copy=rates_total, а затем to_copy=1.

После вычисления размера данных, которые необходимо рассчитать в вызове функции OnCalculate (), производится их вычисление и заполнение ими буферов индикатора.

Если индикатор рассчитывается на основе хэндла другого индикатора, тогда производится копирование из буферов используемого индикатора в динамические массивы данного индикатора.

Вот как это реализовано для используемого индикатора ADX:

// - - заполняем часть массива ADXBuffer значениями из индикаторного буфера под индексом 0

```
if (CopyBuffer (ind_handle,0,0,amount, adx_values) <0)
```

```
{
```

```
// - - если копирование не удалось, сообщим код ошибки
```

```
PrintFormat («Не удалось скопировать данные из индикатора iADX, код ошибки %d», GetLastError ());
```

// - - завершим с нулевым результатом – это означает, что индикатор будет считаться нерассчитанным

```
return (false);
```

```
}
```

// - - заполняем часть массива DI_plusBuffer значениями из индикаторного буфера под индексом 1

```
if (CopyBuffer (ind_handle,1,0,amount, DIplus_values) <0)
```

```
{
```

```
// - - если копирование не удалось, сообщим код ошибки
```

```
PrintFormat («Не удалось скопировать данные из индикатора iADX, код ошибки %d»,  
GetLastError ());  
// - - завершим с нулевым результатом – это означает, что индикатор будет считаться  
нерассчитанным  
return (false);  
}
```

```
// - - заполняем часть массива DI_plusBuffer значениями из индикаторного буфера под  
индексом 2  
if (CopyBuffer (ind_handle,2,0,amount, DIminus_values) <0)  
{  
// - - если копирование не удалось, сообщим код ошибки  
PrintFormat («Не удалось скопировать данные из индикатора iADX, код ошибки %d»,  
GetLastError ());  
// - - завершим с нулевым результатом – это означает, что индикатор будет считаться  
нерассчитанным  
return (false);  
}
```

Здесь `ind_handle` – это хэндл индикатора ADX, второй параметр – индекс буфера используемого индикатора, из которого производится копирование, третий параметр – стартовая позиция, откуда начинается копирование. Здесь мы помним, что копирование идет от конца к началу, и поэтому нулевая стартовая позиция – это самые свежие данные. Четвертый параметр – это наш размер данных, которые необходимо рассчитать в вызове функции `OnCalculate()`, и последний параметр – это обычно динамический массив, привязанный к буферу индикатора, куда производится копирование.

Тут есть вопрос, как связать второй параметр функции `CopyBuffer` с индексом буфера используемого индикатора.

Это определяется вызовом функции `SetIndexBuffer` в используемом индикаторе. Например, для индикатора ADX:

```
SetIndexBuffer (0,ExtADXBuffer);  
SetIndexBuffer (1,ExtPDIBuffer);  
SetIndexBuffer (2,ExtNDIBuffer);
```

Отсюда нулевой индекс связан с буфером самого индикатора ADX, 1 индекс связан с буфером индикатора направленности +DI, 2 индекс связан с буфером индикатора направленности —DI.

Таким образом, для связывания второго параметра функции `CopyBuffer` с индексом буфера используемого индикатора, нужно знать код используемого индикатора.

Также для заполнения буфера индикатора значениями, может использоваться цикл, например:

```
for (int i=start; i < rates_total && !IsStopped (); i++)  
{  
  
}
```

Здесь `start` – это стартовая позиция, с которой начинается заполнение буфера индикатора.

При значении `prev_calculated=0`, значение `start` это, как правило, 0, при значении `prev_calculated= rates_total`, значение `start=prev_calculated-1`.

Если же перед реализацией цикла с помощью функции `ArraySetAsSeries` поменять порядок доступа к массивам буферов индикатора и цен, тогда цикл примет вид:

```
for (int i=start; i>=0;i - ) {
```

```
}
```

Где start=rates_total-1, если prev_calculated=0, и start=0, если prev_calculated=rates_total.

Пример создания индикатора

В качестве примера рассмотрим создание индикатора, который будет реализовывать форекс стратегию «Impulse keeper» (Ловец импульсов) и показывать на графике сигналы на покупку и продажу.

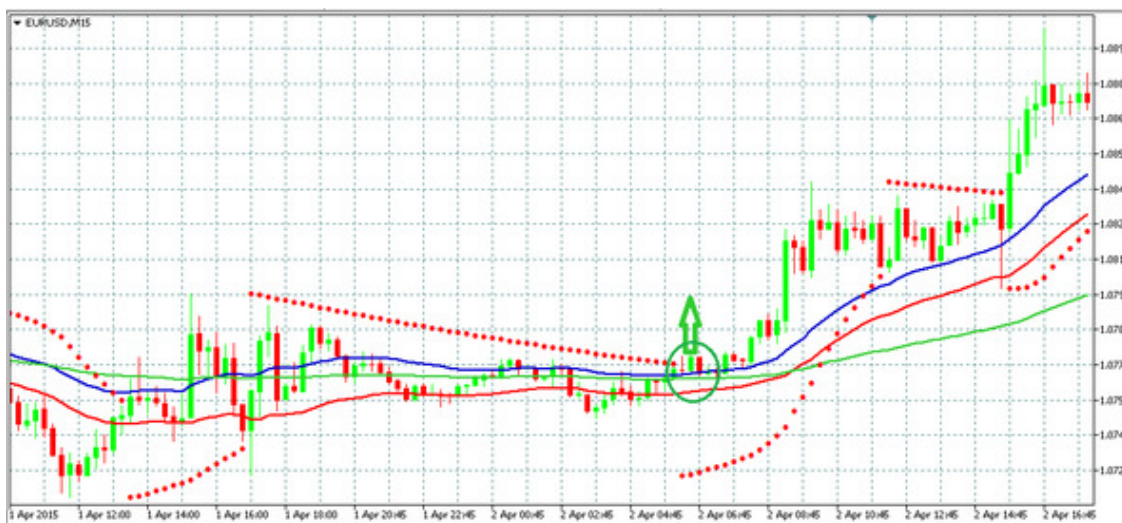
В данной стратегии применяются четыре индикатора:

Экспоненциальная скользящая средняя с периодом 34 для цены High.

Экспоненциальная скользящая средняя с периодом 34 для цены Low.

Экспоненциальная скользящая средняя с периодом 125 для цены Close.

Parabolic SAR.



Сигналы на покупку и продажу в данной стратегии описываются следующим образом.

Сигнал на покупку: зеленая свеча закрывается выше EMA34 High и EMA34 Low, зеленая свеча выше EMA125 и Parabolic SAR.

Сигнал на продажу: красная свеча закрывается ниже EMA34 Low и EMA34 High, красная свеча ниже EMA125 и Parabolic SAR.

Давайте, реализуем эту стратегию в коде, который будет отображать на графике стрелки вверх и вниз сигналов на покупку и продажу.

Откроем MQL5 редактор и в меню File выберем New. В диалоговом окне MQL Wizard выберем Custom Indicator и нажмем кнопку Далее. Введем имя индикатора Impulse keeper, имя автора и ссылку и нажмем два раза Далее, а затем Готово.

В результате мы получим код индикатора с пустыми функциями OnInit и OnCalculate.

Создание индикатора начнем с определения его свойств.

Количество буферов индикатора определим 8.

2 буфера – данные и цвет, для сигналов на покупку. 2 буфера – данные и цвет, для сигналов на продажу. И 4 буфера промежуточных вычислений для скопированных данных из индикаторов EMA34 Low, EMA34 High, EMA125 и Parabolic SAR:

```
#property indicator_buffers 8
```

Определим число графических построений – 2, одно построение для сигналов на покупку и другое построение для сигналов на продажу:

```
#property indicator_plots 2
```

Определим цвет и тип для обоих графических построений:

```
#property indicator_color1 clrGreen, clrBlack
```

```
#property indicator_type1 DRAW_COLOR_ARROW
```

```
#property indicator_color2 clrRed, clrBlack
```

```
#property indicator_type2 DRAW_COLOR_ARROW
```

Далее определим массивы буферов индикатора и хэндлы используемых индикаторов:

```
double IKBuyBuffer [];
```

```
double ColorIKBuyBuffer [];
```

```
double IKSellBuffer [];
```

```
double ColorIKSellBuffer [];
```

```
double EMA34HBuffer [];
```

```
double EMA34LBuffer [];
```

```
double EMA125Buffer [];
```

```
double PSARBuffer [];
```

```
int EMA34HHandle;
```

```
int EMA34LHandle;
```

```
int EMA125Handle;
```

```
int PSARHandle;
```

В функции OnInit () для первого графического построения определим тип стрелки – стрелка вверх, пустое значение и сдвиг:

```
int OnInit ()
```

```
{
```

```
PlotIndexSetInteger (0,PLOT_ARROW,233);
```

```
PlotIndexSetDouble (0,PLOT_EMPTY_VALUE,0);
```

```
PlotIndexSetInteger (0,PLOT_ARROW_SHIFT, -10);
```

Для второго графического построения определим тип стрелки – стрелка вниз, пустое значение и сдвиг:

```
PlotIndexSetInteger (1,PLOT_ARROW,234);
```

```
PlotIndexSetDouble (1,PLOT_EMPTY_VALUE,0);
```

```
PlotIndexSetInteger (1,PLOT_ARROW_SHIFT,10);
```

Свяжем массивы с буферами индикатора:

```
SetIndexBuffer (0,IKBuyBuffer, INDICATOR_DATA);
```

```
SetIndexBuffer (1,ColorIKBuyBuffer, INDICATOR_COLOR_INDEX);
```

```
SetIndexBuffer (2,IKSellBuffer, INDICATOR_DATA);
SetIndexBuffer (3,ColorIKSellBuffer, INDICATOR_COLOR_INDEX);
```

```
SetIndexBuffer (4,EMA34HBuffer, INDICATOR_CALCULATIONS);
SetIndexBuffer (5,EMA34LBuffer, INDICATOR_CALCULATIONS);
SetIndexBuffer (6,EMA125Buffer, INDICATOR_CALCULATIONS);
SetIndexBuffer (7,PSARBuffer, INDICATOR_CALCULATIONS);
```

Получим хэндлы используемых индикаторов:

```
EMA34HHandle=iMA (NULL,0,34,0,MODE_EMA, PRICE_HIGH);
EMA34LHandle=iMA (NULL,0,34,0,MODE_EMA, PRICE_LOW);
EMA125Handle=iMA (NULL,0,125,0,MODE_EMA, PRICE_CLOSE);
PSARHandle=iSAR (NULL,0,0.02, 0.2);
```

В функции OnCalculate () произведем проверку размера доступной истории для расчета используемых индикаторов, определим количество копируемых значений используемых индикаторов и определим стартовую позицию расчета индикатора:

```
int values_to_copy;
int start;
int calculated=BarsCalculated (EMA34HHandle);
if (calculated <=0)
{
return (0);
}
if (prev_calculated==0 || calculated!=bars_calculated)
{
start=1;
if (calculated> rates_total) values_to_copy=rates_total;
else values_to_copy=calculated;
}
else
{
start=rates_total-1;
values_to_copy=1;
}
```

Переменную bars_calculated определим как глобальную int bars_calculated=0; в свойствах индикатора.

Далее произведем копирование из буферов используемых индикаторов в массивы буферов нашего индикатора:

```
if (!FillArrayFromMABuffer (EMA34HBuffer,0,EMA34HHandle, values_to_copy))
return (0); if (!FillArrayFromMABuffer (EMA34LBuffer,0,EMA34LHandle, values_to_copy))
return (0); if (!FillArrayFromMABuffer (EMA125Buffer,0,EMA125Handle, values_to_copy))
return (0);
```

```
if (!FillArrayFromPSARBuffer (PSARBuffer, PSARHandle, values_to_copy)) return (0);
```

Здесь FillArrayFromMABuffer и FillArrayFromPSARBuffer – пользовательские функции, определенные вне функции OnCalculate ():

```
//+-----+
bool FillArrayFromPSARBuffer (
double &sar_buffer [], // индикаторный буфер значений Parabolic SAR
int ind_handle, // хэндл индикатора iSAR
```

```
int amount // количество копируемых значений
)
{
ResetLastError ();
if (CopyBuffer (ind_handle,0,0,amount, sar_buffer) <0)
{
return (false);
}
return (true);
}
//+-----+
bool FillArrayFromMABuffer (
double &values [], // индикаторный буфер значений Moving Average
int shift, // смещение
int ind_handle, // хэндл индикатора iMA
int amount // количество копируемых значений
)
{
ResetLastError ();
if (CopyBuffer (ind_handle,0, -shift, amount, values) <0)
{
return (false);
}
return (true);
}
Далее в функции OnCalculate () заполним буфера индикатора данными и цветом:
for (int i=start; i <rates_total &&!IsStopped ();i++)
{
IKBuyBuffer [i-1] =0;
ColorIKBuyBuffer [i-1] =1;

IKSellBuffer [i-1] =0;
ColorIKSellBuffer [i-1] =1;

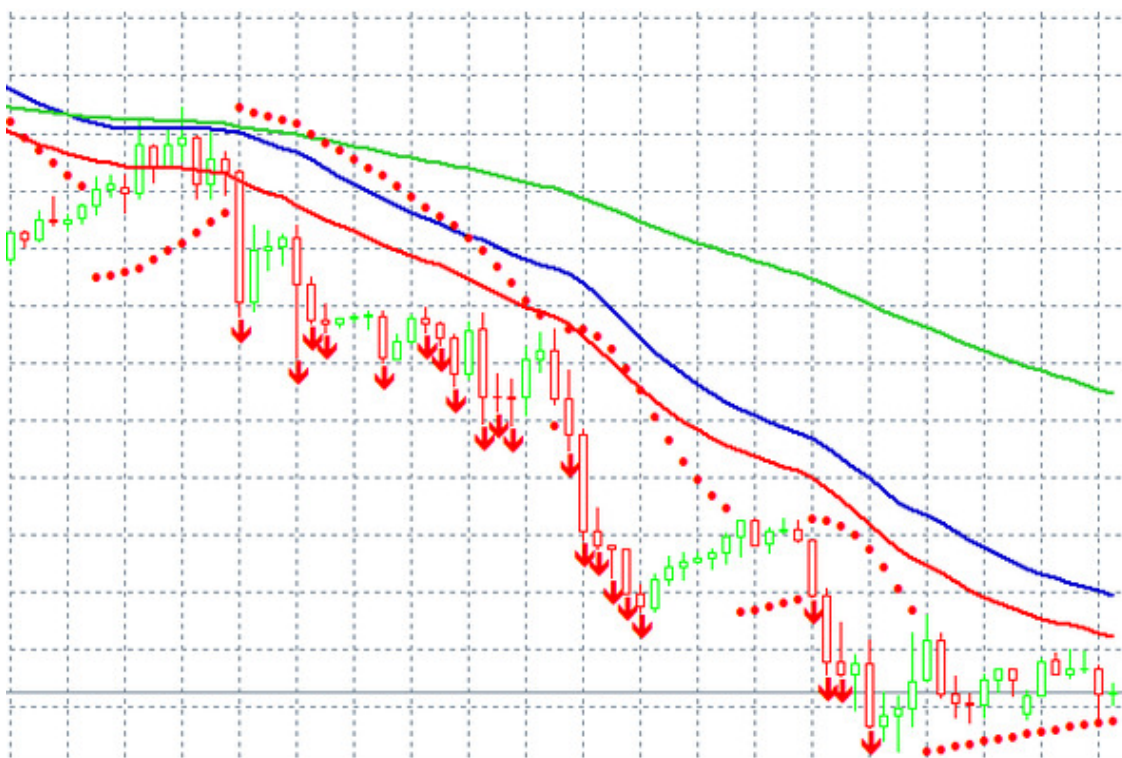
if (close [i-1]> open [i-1] &&close [i-1]> EMA34HBuffer [i-1] &&close [i-1]>
EMA34LBuffer [i-1] &&low [i-1]> EMA125Buffer [i-1] &&low [i-1]> PSARBuffer [i-1]
&&EMA125Buffer [i-1] <EMA34LBuffer [i-1] &&EMA125Buffer [i-1] <EMA34HBuffer
[i-1]) {
IKBuyBuffer [i-1] =high [i-1];
ColorIKBuyBuffer [i-1] =0;
}

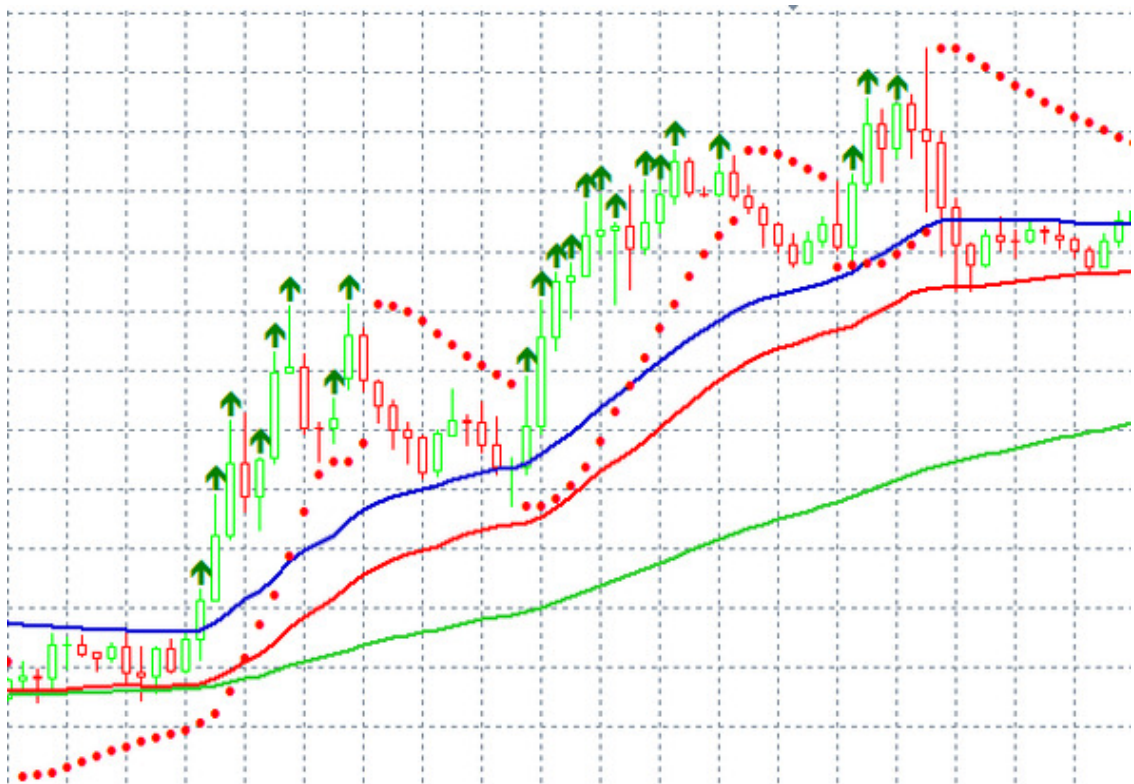
if (close [i-1] <open [i-1] &&close [i-1] <EMA34HBuffer [i-1] &&close [i-1]
<EMA34LBuffer [i-1] &&high [i-1] <EMA125Buffer [i-1] &&high [i-1] <PSARBuffer [i-1]
&&EMA125Buffer [i-1]> EMA34LBuffer [i-1] &&EMA125Buffer [i-1]> EMA34HBuffer
[i-1]) {
IKSellBuffer [i-1] =low [i-1];
ColorIKSellBuffer [i-1] =0;
}
}
```

```
}  
bars_calculated=calculated;  
// - - return value of prev_calculated for next call  
return (rates_total);  
}
```

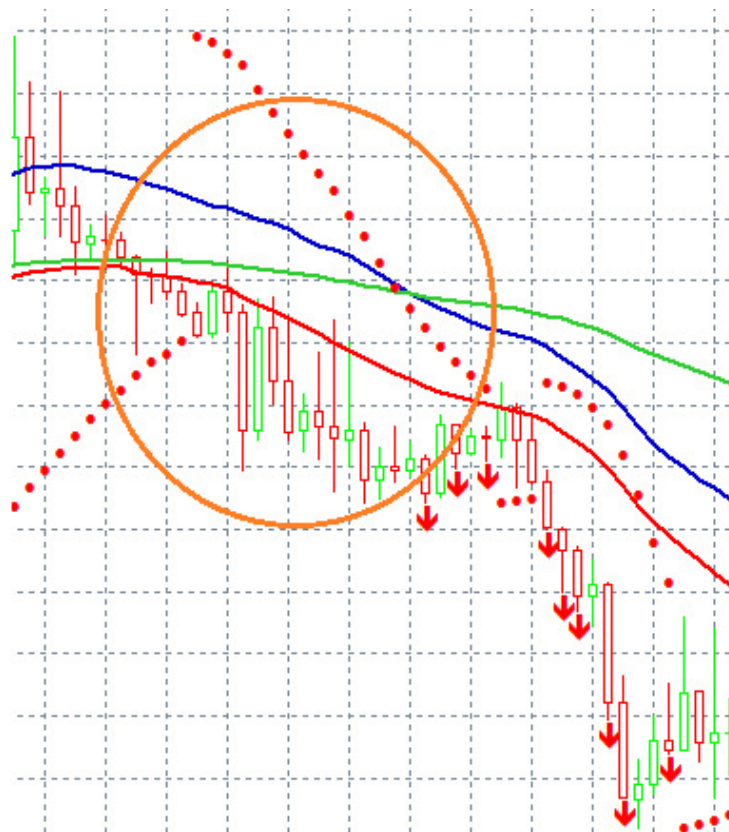
Здесь мы рассчитываем индикатор на предыдущем баре, так как на текущем баре цена close – это текущая цена тика.

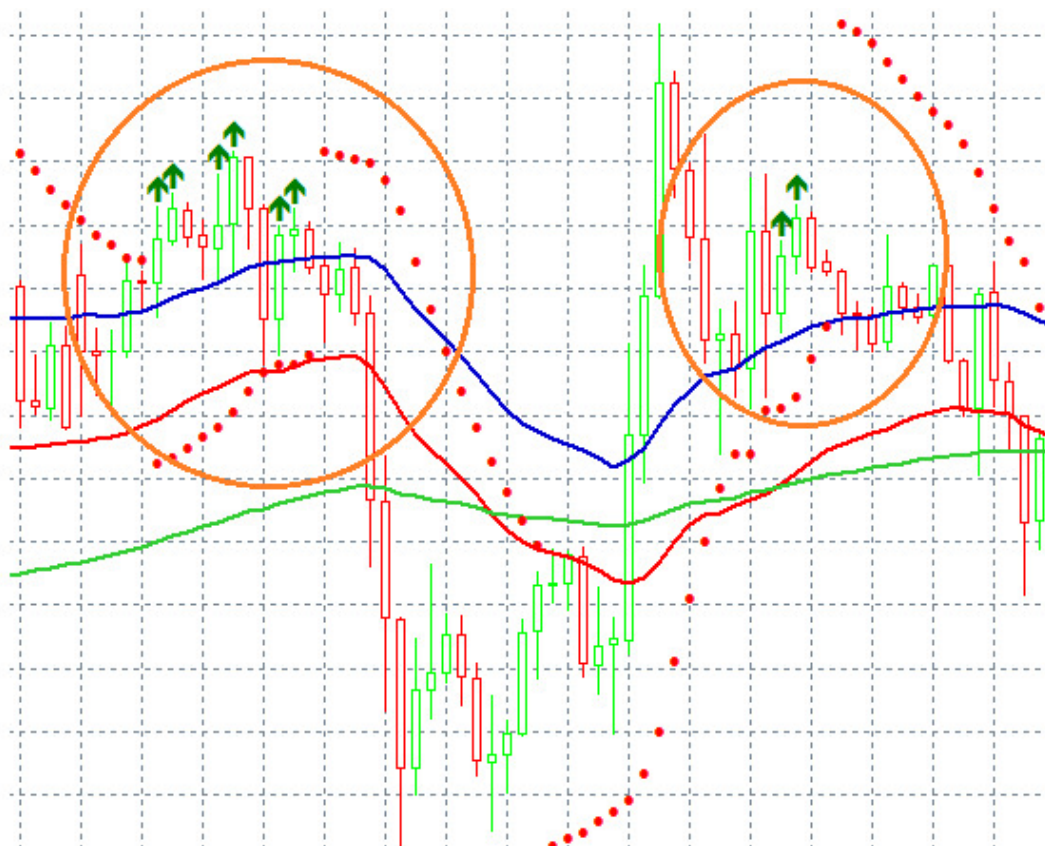
Откомпилируем код и присоединим индикатор к графику:





Мы увидим, что, в общем и целом, индикатор дает верные сигналы на продажу и покупку, хотя в некоторых случаях он запаздывает и дает ложные сигналы:



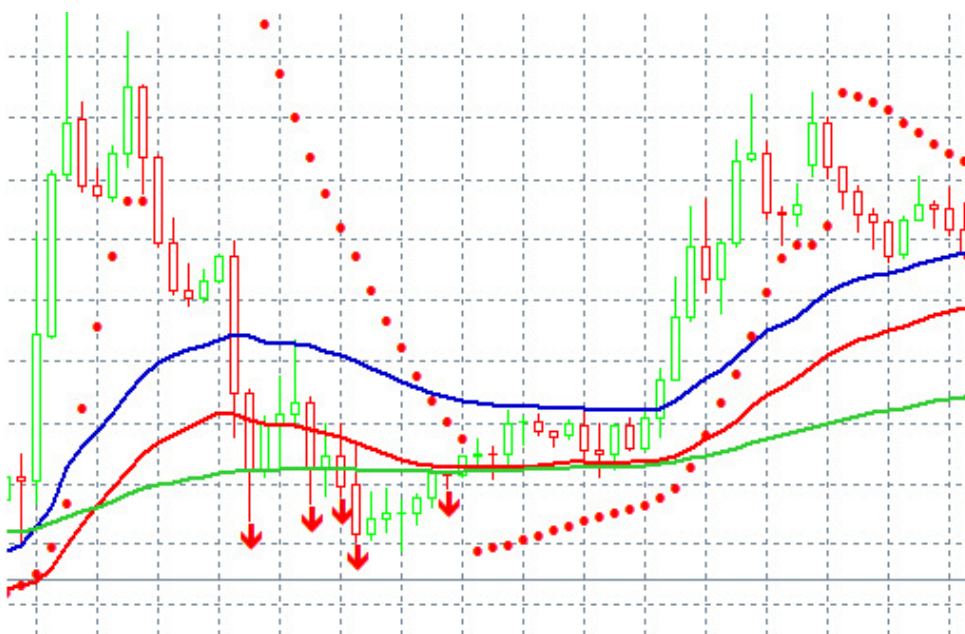
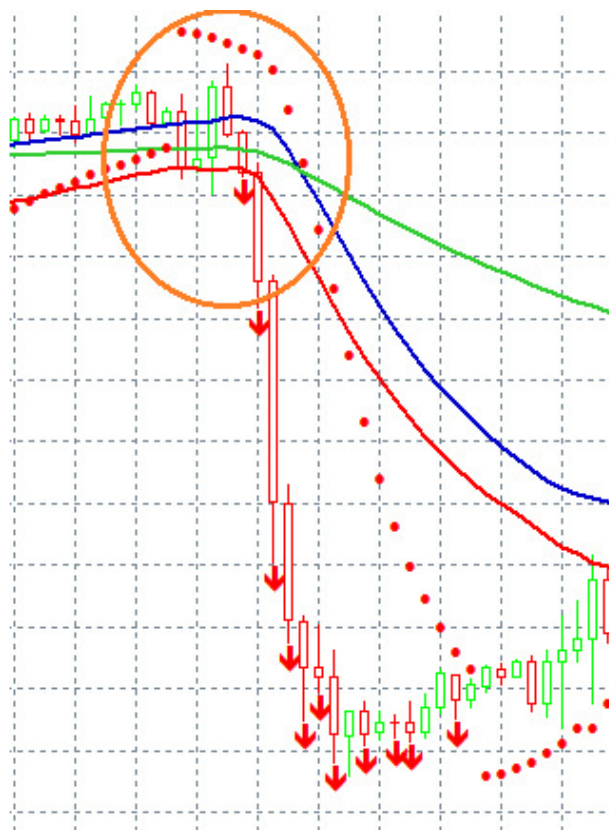


Как мы видим, происходит это из-за трендовой линии EMA125.

Попробуем отвязать ее от текущего периода и попробуем определять тренд, скажем по дневному графику:

```
EMA125Handle=iMA (NULL, PERIOD_D1,125,0,MODE_EMA, PRICE_CLOSE);
```

При этом запаздывание, конечно, сократится, но количество ложных сигналов увеличится:



Видимо для улучшения данной стратегии, нужно привлекать дополнительные индикаторы.

Попробуем сделать этот же самый индикатор, но не с помощью графических построений, а помещая графические объекты на график символа.

В свойствах индикатора теперь не нужно объявлять буфера данных и цвета индикатора и графические серии для них. Оставим только буфера индикатора для промежуточных расчетов и хэндлы используемых индикаторов:

```
#property indicator_chart_window
```

```
#property indicator_buffers 4
double EMA34HBuffer [];
double EMA34LBuffer [];
double EMA125Buffer [];
double PSARBuffer [];
int EMA34HHandle;
int EMA34LHandle;
int EMA125Handle;
int PSARHandle;
int bars_calculated=0;
```

В функции OnInit () соответственно оставим только привязку массивов к буферам промежуточных расчетов и получение хэндлов используемых индикаторов:

```
int OnInit ()
{
// - - indicator buffers mapping
SetIndexBuffer (0,EMA34HBuffer, INDICATOR_CALCULATIONS);
SetIndexBuffer (1,EMA34LBuffer, INDICATOR_CALCULATIONS);
SetIndexBuffer (2,EMA125Buffer, INDICATOR_CALCULATIONS);
SetIndexBuffer (3,PSARBuffer, INDICATOR_CALCULATIONS);

EMA34HHandle=iMA (NULL,0,34,0,MODE_EMA, PRICE_HIGH);
EMA34LHandle=iMA (NULL,0,34,0,MODE_EMA, PRICE_LOW);
EMA125Handle=iMA (NULL,0,125,0,MODE_EMA, PRICE_CLOSE);
PSARHandle=iSAR (NULL,0,0.02, 0.2);
// - -
return (INIT_SUCCEEDED);
}
```

В функции OnCalculate () определим создание объектов на графике символа:

```
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 values_to_copy;
int start;
int calculated=BarsCalculated (EMA34HHandle);
if (calculated <=0)
{
return (0);
}
if (prev_calculated==0 || calculated!=bars_calculated)
{
```

```
start=1;
if (calculated> rates_total) values_to_copy=rates_total;
else values_to_copy=calculated;
}
else
{
start=rates_total-1;
values_to_copy=1;
} if (!FillArrayFromMABuffer (EMA34HBuffer,0,EMA34HHandle, values_to_copy))
return (0); if (!FillArrayFromMABuffer (EMA34LBuffer,0,EMA34LHandle, values_to_copy))
return (0); if (!FillArrayFromMABuffer (EMA125Buffer,0,EMA125Handle, values_to_copy))
return (0);
if (!FillArrayFromPSARBuffer (PSARBuffer, PSARHandle, values_to_copy)) return (0);

for (int i=start; i <rates_total &&!IsStopped ();i++)
{
if (close [i-1]> open [i-1] &&close [i-1]> EMA34HBuffer [i-1] &&close [i-1]>
EMA34LBuffer [i-1] &&low [i-1]> EMA125Buffer [i-1] &&low [i-1]> PSARBuffer [i-1]
&&EMA125Buffer [i-1] <EMA34LBuffer [i-1] &&EMA125Buffer [i-1] <EMA34HBuffer
[i-1]) {
if (!ObjectCreate (0,«Buy»+i, OBJ_ARROW,0,time [i-1],high [i-1]))
{
return (false);
}
ObjectSetInteger (0,«Buy»+i, OBJPROP_COLOR, clrGreen);
ObjectSetInteger (0,«Buy»+i, OBJPROP_ARROWCODE,233);
ObjectSetInteger (0,«Buy»+i, OBJPROP_WIDTH,2);
ObjectSetInteger (0,«Buy»+i, OBJPROP_ANCHOR, ANCHOR_UPPER);
ObjectSetInteger (0,«Buy»+i, OBJPROP_HIDDEN, true);
ObjectSetString (0,«Buy»+i, OBJPROP_TOOLTIP,»»+close [i-1]);
}
if (close [i-1] <open [i-1] &&close [i-1] <EMA34HBuffer [i-1] &&close [i-1]
<EMA34LBuffer [i-1] &&high [i-1] <EMA125Buffer [i-1] &&high [i-1] <PSARBuffer [i-1]
&&EMA125Buffer [i-1]> EMA34LBuffer [i-1] &&EMA125Buffer [i-1]> EMA34HBuffer
[i-1]) {
if (!ObjectCreate (0,«Sell»+i, OBJ_ARROW,0,time [i-1],low [i-1]))
{
return (false);
}
ObjectSetInteger (0,«Sell»+i, OBJPROP_COLOR, clrRed);
ObjectSetInteger (0,«Sell»+i, OBJPROP_ARROWCODE,234);
ObjectSetInteger (0,«Sell»+i, OBJPROP_WIDTH,2);
ObjectSetInteger (0,«Sell»+i, OBJPROP_ANCHOR, ANCHOR_LOWER);
ObjectSetInteger (0,«Sell»+i, OBJPROP_HIDDEN, true);
ObjectSetString (0,«Sell»+i, OBJPROP_TOOLTIP,»»+close [i-1]);
}
}
bars_calculated=calculated;
// - - return value of prev_calculated for next call
```

```
return (rates_total);  
}
```

Здесь функцией `ObjectCreate` создаются объекты стрелка, привязанные ко времени и максимальной или минимальной цене.

Функцией `ObjectSetInteger` со свойством `OBJPROP_COLOR` определяется цвет стрелки.

Функцией `ObjectSetInteger` со свойством `OBJPROP_ARROWCODE` определяется направление стрелки вверх или вниз.

Функцией `ObjectSetInteger` со свойством `OBJPROP_WIDTH` определяется размер объекта.

Функцией `ObjectSetInteger` со свойством `OBJPROP_ANCHOR` определяется привязка к цене сверху или снизу по центру.

Функцией `ObjectSetInteger` со свойством `OBJPROP_HIDDEN` – `true` определяется отсутствие созданных объектов в списке объектов графика символа.

Функцией `ObjectSetString` со свойством `OBJPROP_TOOLTIP` определяется содержание всплывающей подсказки при наведении указателя на объект.

В функции `OnDeinit ()` уберем все добавленные графические объекты:

```
void OnDeinit (const int reason) {  
    ObjectsDeleteAll (0, -1, -1);  
}
```

Более подробно о создании объектов на графике символа мы поговорим далее.

Графические объекты

Как уже было показано ранее, мы можем рисовать на графике символа не только диаграммы индикатора, но и добавлять различные графические объекты с помощью функции `ObjectCreate`:

```
bool ObjectCreate (  
    long chart_id, // идентификатор графика  
    string name, // имя объекта  
    ENUM_OBJECT type, // тип объекта  
    int sub_window, // индекс окна  
    datetime time1, // время первой точки привязки  
    double price1, // цена первой точки привязки
```

```
    datetime timeN=0, // время N-ой точки привязки  
    double priceN=0, // цена N-ой точки привязки
```

```
    datetime time30=0, // время 30-й точки привязки  
    double price30=0 // цена 30-точки привязки
```

```
);
```

Здесь параметр `sub_window` это индекс главного окна графика символа со значением 0 или индекс подокна другого индикатора, присоединенного к графику символа.

Например, если в предыдущем примере мы изменим код, и присоединим к графику символа, скажем, индикатор ADX, мы увидим следующее:

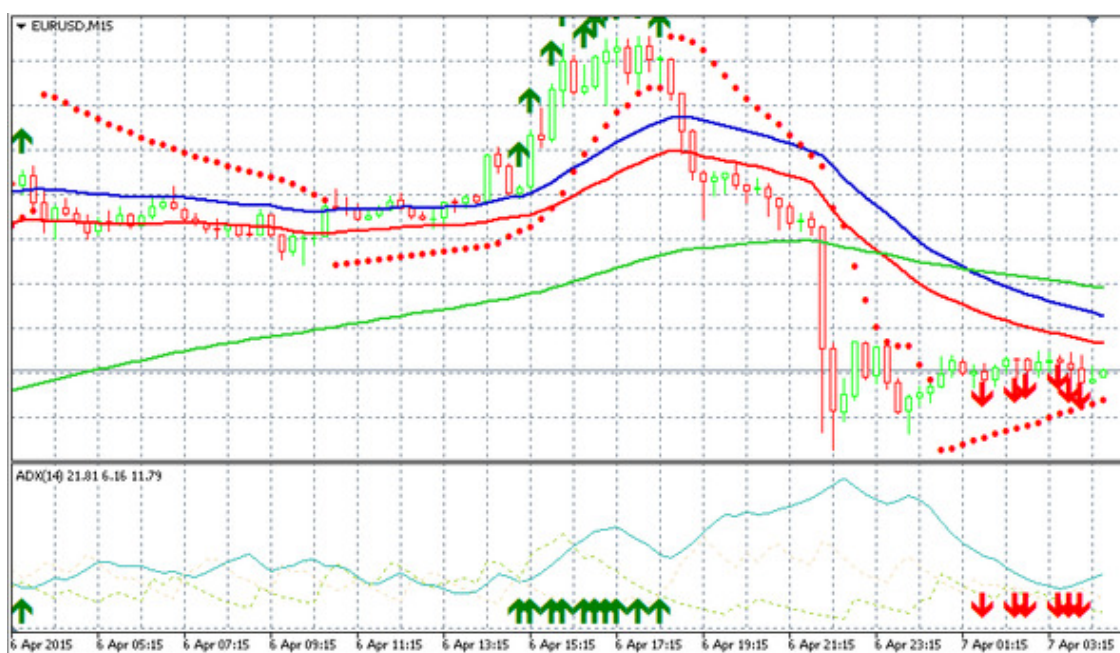
```
    for (int i=start; i < rates_total && !IsStopped (); i++)  
    {  
        if (close [i-1] > open [i-1] && close [i-1] > EMA34HBuffer [i-1] && close [i-1] >  
            EMA34LBuffer [i-1] && low [i-1] > EMA125Buffer [i-1] && low [i-1] > PSARBuffer [i-1]  
            && EMA125Buffer [i-1] < EMA34LBuffer [i-1] && EMA125Buffer [i-1] < EMA34HBuffer  
            [i-1]) {  
            if (!ObjectCreate (0, «Buy»+i, OBJ_ARROW, 0, time [i-1], high [i-1]))  
            {  
                return (false);  
            }  
            if (!ObjectCreate (0, «Buy1»+i, OBJ_ARROW, 1, time [i-1], high [i-1]))  
            {  
                return (false);  
            }  
            ObjectSetInteger (0, «Buy»+i, OBJPROP_COLOR, clrGreen);  
            ObjectSetInteger (0, «Buy»+i, OBJPROP_ARROWCODE, 233);  
            ObjectSetInteger (0, «Buy»+i, OBJPROP_WIDTH, 2);  
            ObjectSetInteger (0, «Buy»+i, OBJPROP_ANCHOR, ANCHOR_UPPER);  
            ObjectSetInteger (0, «Buy»+i, OBJPROP_HIDDEN, true);  
            ObjectSetString (0, «Buy»+i, OBJPROP_TOOLTIP, close [i-1]);  
  
            ObjectSetInteger (0, «Buy1»+i, OBJPROP_COLOR, clrGreen);  
            ObjectSetInteger (0, «Buy1»+i, OBJPROP_ARROWCODE, 233);  
            ObjectSetInteger (0, «Buy1»+i, OBJPROP_WIDTH, 2);  
            ObjectSetInteger (0, «Buy1»+i, OBJPROP_ANCHOR, ANCHOR_UPPER);
```

```

ObjectSetInteger (0,«Buy1»+i, OBJPROP_HIDDEN, true);
ObjectSetString (0,«Buy1»+i, OBJPROP_TOOLTIP, close [i-1]);
}
if (close [i-1] <open [i-1] &&close [i-1] <EMA34HBuffer [i-1] &&close [i-1]
<EMA34LBuffer [i-1] &&high [i-1] <EMA125Buffer [i-1] &&high [i-1] <PSARBuffer [i-1]
&&EMA125Buffer [i-1]> EMA34LBuffer [i-1] &&EMA125Buffer [i-1]> EMA34HBuffer
[i-1]) {
    if (!ObjectCreate (0,«Sell»+i, OBJ_ARROW,0,time [i-1],low [i-1]))
    {
        return (false);
    }
    if (!ObjectCreate (0,«Sell1»+i, OBJ_ARROW,1,time [i-1],low [i-1]))
    {
        return (false);
    }
    ObjectSetInteger (0,«Sell»+i, OBJPROP_COLOR, clrRed);
    ObjectSetInteger (0,«Sell»+i, OBJPROP_ARROWCODE,234);
    ObjectSetInteger (0,«Sell»+i, OBJPROP_WIDTH,2);
    ObjectSetInteger (0,«Sell»+i, OBJPROP_ANCHOR, ANCHOR_LOWER);
    ObjectSetInteger (0,«Sell»+i, OBJPROP_HIDDEN, true);
    ObjectSetString (0,«Sell»+i, OBJPROP_TOOLTIP, close [i-1]);

    ObjectSetInteger (0,«Sell1»+i, OBJPROP_COLOR, clrRed);
    ObjectSetInteger (0,«Sell1»+i, OBJPROP_ARROWCODE,234);
    ObjectSetInteger (0,«Sell1»+i, OBJPROP_WIDTH,2);
    ObjectSetInteger (0,«Sell1»+i, OBJPROP_ANCHOR, ANCHOR_LOWER);
    ObjectSetInteger (0,«Sell1»+i, OBJPROP_HIDDEN, true);
    ObjectSetString (0,«Sell1»+i, OBJPROP_TOOLTIP, close [i-1]);
}
}

```



Нумерация подокон идет сверху вниз в порядке отображения.

Тип отображаемого объекта задается перечислением ENUM_OBJECT, которое можно посмотреть в справочнике.

После добавления графических объектов, не забываем их удалять в функции обратного вызова OnDeinit (), используя функцию ObjectDelete:

```
bool ObjectDelete (  
    long chart_id, // chart identifier  
    string name // object name  
);
```

Или используя функцию ObjectsDeleteAll:

```
int ObjectsDeleteAll (  
    long chart_id, // chart identifier  
    int sub_window=-1, // window index  
    int type=-1 // object type  
);
```

Помимо вышеупомянутых функций ObjectCreate, ObjectDelete и ObjectsDeleteAll, MQL5 предлагает набор функций для работы с графическими объектами: ObjectName, ObjectFind, ObjectGetTimeByValue, ObjectGetValueByTime, ObjectMove, ObjectsTotal, ObjectGetDouble, ObjectGetInteger, ObjectGetString, ObjectSetDouble, ObjectSetInteger, ObjectSetString, TextSetFont, TextOut, TextGetSize.

Функции ObjectName, ObjectFind, ObjectGetTimeByValue, ObjectGetValueByTime, ObjectsTotal, ObjectGetDouble, ObjectGetInteger, ObjectGetString, TextGetSize – это функции возвращающие информацию.

Функции ObjectSetDouble, ObjectSetInteger, ObjectSetString, TextSetFont – это функции устанавливающие свойства объекта.

Функция ObjectMove перемещает объект в окне.

Функция TextOut выводит текст в пиксельный массив для отображения объектом OBJ_BITMAP_LABEL или OBJ_BITMAP.

После добавления графических объектов рекомендуется принудительно перерисовать график символа с помощью функции ChartRedraw:

```
void ChartRedraw (  
    long chart_id=0 // идентификатор графика  
);
```

Функция ObjectCreate позволяет создавать программным способом те графические объекты, которые вы можете вручную нарисовать на графике символа, пользуясь панелью инструментов клиентского терминала.

С помощью функции ObjectSetDouble устанавливаются такие свойства графического объекта, как OBJPROP_PRICE – изменение параметра price функции ObjectCreate, OBJPROP_LEVELVALUE – определение уровней для таких объектов, как инструменты Фибоначи и Вилы Эндрюса, OBJPROP_SCALE – определение масштаба для таких объектов, как инструменты Ганна и Дуги Фибоначчи, OBJPROP_ANGLE – определение угла объекта, т.е. возможность повернуть объект, который изначально не имеет жесткой привязки, например, повернуть текст, OBJPROP_DEVIATION – определение отклонения для объекта Канал стандартного отклонения.

Пример использования OBJPROP_PRICE:

```
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, true);  
    ArraySetAsSeries (high, true);  
    ArraySetAsSeries (low, true);  
    ArraySetAsSeries (close, true);  
    ObjectDelete (0,«Price»);  
    if (!ObjectCreate (0,«Price», OBJ_HLINE,0,time [1],close [1]))  
    {  
        return (false);  
    }  
    ObjectSetInteger (0,«Price», OBJPROP_COLOR, clrGreen);  
    ObjectSetInteger (0,«Price», OBJPROP_WIDTH,1);  
    ObjectSetString (0,«Price», OBJPROP_TOOLTIP, close [1]);  
    if (open [1]> close [1])  
        ObjectSetDouble (0,«Price», OBJPROP_PRICE, low [1]);  
    if (open [1] <close [1])  
        ObjectSetDouble (0,«Price», OBJPROP_PRICE, high [1]);  
    // — - return value of prev_calculated for next call  
    return (rates_total);  
}  
//+-----+  
void OnDeinit (const int reason) {  
    ObjectsDeleteAll (0, -1, -1);  
}
```

В этом коде создается горизонтальный уровень, показывающий минимальную или максимальную цену предыдущего бара, в зависимости от того, является ли этот бар бычьим или медвежьим.

Пример использования OBJPROP ANGLE:

```
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, true);
```

```

ArraySetAsSeries (high, true);
ArraySetAsSeries (low, true);
ArraySetAsSeries (close, true);
ObjectDelete (0,«Line»);
ObjectDelete (0,«Price»);
if (!ObjectCreate (0,«Line», OBJ_VLINE,0,time [1],close [1]))
{
return (false);
}
ObjectSetInteger (0,«Line», OBJPROP_COLOR, clrBlue);
ObjectSetInteger (0,«Line», OBJPROP_WIDTH,1);
ObjectSetString (0,«Line», OBJPROP_TOOLTIP, close [1]);

if (!ObjectCreate (0,«Price», OBJ_TEXT,0,time [3],high [1]))
{
return (false);
}
ObjectSetString (0,«Price», OBJPROP_TEXT, close [1]);
ObjectSetInteger (0,«Price», OBJPROP_COLOR, clrBlack);
ObjectSetDouble (0,«Price», OBJPROP_ANGLE,90);
ObjectSetString (0,«Price», OBJPROP_TOOLTIP, close [1]);

// - - return value of prev_calculated for next call
return (rates_total);
}
//+-----+
void OnDeinit (const int reason) {
ObjectsDeleteAll (0, -1, -1);
}

```

Этот код создает вертикальную линию с подписью цены закрытия предыдущего бара.

С помощью функции `ObjectSetInteger` устанавливаются такие свойства графического объекта, как цвет, стиль, размер и др.

С помощью функции `ObjectSetString` можно изменить имя объекта, при этом объект со старым именем будет удален и будет создан объект с новым именем, установить текст для таких объектов, как текст, кнопка, метка, поле ввода, событие, установить текст всплывающей подсказки для объекта, описание уровня для объектов, имеющих уровни, шрифт, имя BMP-файла для объекта «Графическая метка» и «Рисунок», символ для объекта «График».

Функция `TextSetFont` позволяет установить тип шрифта текста, его размер, стиль и угол наклона для объектов, содержащих текст.

Как уже было сказано, функция `TextOut` позволяет скомбинировать текст и изображение. Например, следующий код выводит текст в изображение, залитое одним цветом:

```

uint ExtImg [10000];
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit ()
{
ObjectCreate (0,«Image», OBJ_BITMAP_LABEL,0,0,0);
ObjectSetString (0,«Image», OBJPROP_BMPFILE,»:: IMG»);

```

```

        ArrayFill (ExtImg,0,10000,0xffffffff);
        TextOut          («Text»,          10,10,TA_LEFT|TA_TOP,
ExtImg,100,100,0x000000,COLOR_FORMAT_XRGB_NOALPHA);
        ResourceCreate          («::          IMG»,
ExtImg,100,100,0,0,0,COLOR_FORMAT_XRGB_NOALPHA);
        ChartRedraw ();
        // — —
        return (INIT_SUCCEEDED);
    }

```

Здесь ExtImg это пиксельный массив, представляющий изображение 100х100 пикселей.

Функция ObjectCreate создает объект «Графическая метка», а функция ObjectSetString устанавливает для этого объекта файл изображения с именем::IMG. По поводу знака «::» справочник говорит следующее:

Для использования своего ресурса в коде нужно перед именем ресурса добавлять специальный признак "::.».

Функция ArrayFill заполняет пиксельный массив пикселями белого цвета.

Функция TextOut выводит в пиксельный массив слово «Text».

Функция ResourceCreate создает из пиксельного массива ресурс с именем::IMG.

В итоге на белом фоне отображается надпись «Text».

Также можно вывести текст на готовое изображение:

```

#resource "\\Images\\image.bmp»
uint ExtImg [10000];
//+ ————— +
//| Custom indicator initialization function |
//+ ————— +
int OnInit ()
{
    ObjectCreate (0,«Image», OBJ_BITMAP_LABEL,0,0,0);
    ObjectSetString (0,«Image», OBJPROP_BMPFILE,»:: IMG»);
    uint width=100;
    uint height=100;
    ResourceReadImage("::Images\\image.bmp», ExtImg, width, height);
    TextOut          («Text»,          10,10,TA_LEFT|TA_TOP,          ExtImg,100,100,0xffffffff,
COLOR_FORMAT_XRGB_NOALPHA);
    ResourceCreate          («::          IMG»,
ExtImg,100,100,0,0,0,COLOR_FORMAT_XRGB_NOALPHA);
    ChartRedraw ();
    // — —
    return (INIT_SUCCEEDED);
}

```

Здесь функция ResourceReadImage считывает существующее изображение из папки Images окна Navigator редактора MQL5 в пиксельный массив::IMG, связанный с объектом «Графическая метка», а функция TextOut выводит в пиксельный массив слово «Text».

То же самое можно проделать и с объектом «Рисунок»:

```

#resource "\\Images\\image.bmp»
uint ExtImg [10000];
//+ ————— +
//| Custom indicator initialization function |

```

```

//+-----+
int OnInit ()
{
//-----
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, true);
ArraySetAsSeries (high, true);
ArraySetAsSeries (low, true);
ArraySetAsSeries (close, true);
ObjectDelete (0,«Image»);
ObjectCreate (0,«Image», OBJ_BITMAP,0,time [1],close [1]);
ObjectSetString (0,«Image», OBJPROP_BMPFILE,»:: IMG»);
uint width=100;
uint height=100;
ResourceReadImage("::Images\\image.bmp», ExtImg, width, height);
TextOut («Text», 10,10,TA_LEFT|TA_TOP, ExtImg,100,100,0xffffffff,
COLOR_FORMAT_XRGB_NOALPHA);
ResourceCreate (»:: IMG»,
ExtImg,100,100,0,0,COLOR_FORMAT_XRGB_NOALPHA);
ChartRedraw ();
// - - return value of prev_calculated for next call
return (rates_total);
}
//+-----+
void OnDeinit (const int reason) {
ObjectsDeleteAll (0, -1, -1);
}

```

В качестве примера использования графических объектов, рассмотрим создание индикатора, который выводит в небольшое окно на графике символа тот же график, но с другим временным периодом.

Для этого используем графический объект OBJ_CHART.

В качестве входных параметров индикатора используем символ графика и его период:

#property indicator_chart_window

```
input string InpSymbol=«EURUSD»; // Символ
input ENUM_TIMEFRAMES InpPeriod=PERIOD_CURRENT; // Период
В функции OnInit () создадим графический объект График:
int OnInit ()
{
if (!ObjectCreate (0,«Chart», OBJ_CHART,0,0,0))
{
return (false);
}
}
```

По умолчанию точка привязки этого объекта – левый верхний угол графика.

Определим отступ точки привязки объекта, его размеры, символ и период графика, отображение шкалы времени, размер точки привязки, с помощью которой можно перемещать объект, отображение ценовой шкалы, режим перемещения мышкой, цвет рамки графика:

```
ObjectSetInteger (0,«Chart», OBJPROP_XDISTANCE,10);
ObjectSetInteger (0,«Chart», OBJPROP_YDISTANCE,20);
ObjectSetInteger (0,«Chart», OBJPROP_XSIZE,300);
ObjectSetInteger (0,«Chart», OBJPROP_YSIZE,200);
ObjectSetString (0,«Chart», OBJPROP_SYMBOL, InpSymbol);
ObjectSetInteger (0,«Chart», OBJPROP_PERIOD, InpPeriod);
ObjectSetInteger (0,«Chart», OBJPROP_DATE_SCALE, true);
ObjectSetInteger (0,«Chart», OBJPROP_WIDTH,1);
ObjectSetInteger (0,«Chart», OBJPROP_PRICE_SCALE, true);
ObjectSetInteger (0,«Chart», OBJPROP_SELECTABLE, true);
ObjectSetInteger (0,«Chart», OBJPROP_SELECTED, true);
ObjectSetInteger (0,«Chart», OBJPROP_COLOR, clrBlue);
```

С помощью свойства объектов OBJPROP_CHART_ID функции ObjectGetInteger получим идентификатор графика, используя который мы теперь можем применять функции работы с графиками (https://www.mql5.com/ru/docs/chart_operations) и свойства графиков (https://www.mql5.com/ru/docs/constants/chartconstants/enum_chart_property):

```
long chartId=ObjectGetInteger (0,«Chart», OBJPROP_CHART_ID);
```

Откроем наш график символа, к которому мы хотим присоединить индикатор, и нажав правой кнопкой мышки, выберем пункт в контекстном меню Шаблоны и Сохранить шаблон.

Теперь мы можем перенести на наш графический объект все настройки и индикаторы графика символа:

```
ChartApplyTemplate(chartId,"my.tpl");
ChartRedraw (chartId);
// — —
return (INIT_SUCCEEDED);
}
```

Присоединив индикатор к графику символа, мы можем нажать на нем правой кнопкой мышки и изменить его свойства, включая его период, размеры и др.

Функция PlaySound

Функция PlaySound воспроизводит звуковой файл. Например, это можно делать при появлении сигнала индикатора для напоминания:

```
bool PlaySound (  
    string filename // имя WAV-файла  
);
```

В качестве примера добавим звуковой сигнал в наш индикатор Impulse keeper при появлении первого сигнала на покупку или продажу.

Скачаем какой-нибудь WAV-сигнал из Интернета и поместим его файл в папку Sounds терминала.

Добавим код в индикатор Impulse keeper:

```
#property indicator_chart_window  
#property indicator_buffers 4
```

```
double EMA34HBuffer [];  
double EMA34LBuffer [];  
double EMA125Buffer [];  
double PSARBuffer [];
```

```
int EMA34HHandle;  
int EMA34LHandle;  
int EMA125Handle;  
int PSARHandle;
```

```
int bars_calculated=0;
```

```
int countBuy=0
```

```
int countSell=0;
```

```
//+-----+  
//| Custom indicator initialization function |  
//+-----+  
int OnInit ()
```

```
{  
    // - - indicator buffers mapping  
    SetIndexBuffer (0,EMA34HBuffer, INDICATOR_CALCULATIONS);  
    SetIndexBuffer (1,EMA34LBuffer, INDICATOR_CALCULATIONS);  
    SetIndexBuffer (2,EMA125Buffer, INDICATOR_CALCULATIONS);  
    SetIndexBuffer (3,PSARBuffer, INDICATOR_CALCULATIONS);
```

```
    EMA34HHandle=iMA (NULL,0,34,0,MODE_EMA, PRICE_HIGH);  
    EMA34LHandle=iMA (NULL,0,34,0,MODE_EMA, PRICE_LOW);  
    EMA125Handle=iMA (NULL,0,125,0,MODE_EMA, PRICE_CLOSE);  
    PSARHandle=iSAR (NULL,0,0.02, 0.2);
```

```
    // - -  
    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 [])
{
// --
int values_to_copy;
int start;

int calculated=BarsCalculated (EMA34HHandle);
if (calculated <=0)
{
return (0);
}

if (prev_calculated==0 || calculated!=bars_calculated)
{
start=1;
if (calculated> rates_total) values_to_copy=rates_total;
else values_to_copy=calculated;
}
else
{
start=rates_total-1;
values_to_copy=1;
}
if (!FillArrayFromMABuffer (EMA34HBuffer,0,EMA34HHandle, values_to_copy))
return (0); if (!FillArrayFromMABuffer (EMA34LBuffer,0,EMA34LHandle, values_to_copy))
return (0); if (!FillArrayFromMABuffer (EMA125Buffer,0,EMA125Handle, values_to_copy))
return (0);
if (!FillArrayFromPSARBuffer (PSARBuffer, PSARHandle, values_to_copy)) return (0);

for (int i=start; i <rates_total &&!IsStopped ();i++)
{

if (close [i-1]> open [i-1] &&close [i-1]> EMA34HBuffer [i-1] &&close [i-1]>
EMA34LBuffer [i-1] &&low [i-1]> EMA125Buffer [i-1] &&low [i-1]> PSARBuffer [i-1]
&&EMA125Buffer [i-1] <EMA34LBuffer [i-1] &&EMA125Buffer [i-1] <EMA34HBuffer
[i-1]) {
if (!ObjectCreate (0,«Buy»+i, OBJ_ARROW,0,time [i-1],high [i-1]))
{

```

```
return (false);
}
ObjectSetInteger (0,«Buy»+i, OBJPROP_COLOR, clrGreen);
ObjectSetInteger (0,«Buy»+i, OBJPROP_ARROWCODE,233);
ObjectSetInteger (0,«Buy»+i, OBJPROP_WIDTH,2);
ObjectSetInteger (0,«Buy»+i, OBJPROP_ANCHOR, ANCHOR_UPPER);
ObjectSetInteger (0,«Buy»+i, OBJPROP_HIDDEN, true);
ObjectSetString (0,«Buy»+i, OBJPROP_TOOLTIP, close [i-1]);
}

if (start!=1) {
    if (close [i-1]> open [i-1] &&close [i-1]> EMA34HBuffer [i-1] &&close [i-1]>
EMA34LBuffer [i-1] &&low [i-1]> EMA125Buffer [i-1] &&low [i-1]> PSARBuffer [i-1]
&&EMA125Buffer [i-1] <EMA34LBuffer [i-1] &&EMA125Buffer [i-1] <EMA34HBuffer
[i-1]) {
        countBuy++;
        if (countBuy==1) PlaySound («chime. wav»)
        } else {
            countBuy=0;
        }

        if (close [i-1] <open [i-1] &&close [i-1] <EMA34HBuffer [i-1] &&close [i-1]
<EMA34LBuffer [i-1] &&high [i-1] <EMA125Buffer [i-1] &&high [i-1] <PSARBuffer [i-1]
&&EMA125Buffer [i-1]> EMA34LBuffer [i-1] &&EMA125Buffer [i-1]> EMA34HBuffer
[i-1]) {
            countSell++;
            if (countSell==1) PlaySound («chime. wav»);
            } else {
                countSell=0;
            }
        }

        if (close [i-1] <open [i-1] &&close [i-1] <EMA34HBuffer [i-1] &&close [i-1]
<EMA34LBuffer [i-1] &&high [i-1] <EMA125Buffer [i-1] &&high [i-1] <PSARBuffer [i-1]
&&EMA125Buffer [i-1]> EMA34LBuffer [i-1] &&EMA125Buffer [i-1]> EMA34HBuffer
[i-1]) {
            if (!ObjectCreate (0,«Sell»+i, OBJ_ARROW,0,time [i-1],low [i-1]))
            {
                return (false);
            }
            ObjectSetInteger (0,«Sell»+i, OBJPROP_COLOR, clrRed);
            ObjectSetInteger (0,«Sell»+i, OBJPROP_ARROWCODE,234);
            ObjectSetInteger (0,«Sell»+i, OBJPROP_WIDTH,2);
            ObjectSetInteger (0,«Sell»+i, OBJPROP_ANCHOR, ANCHOR_LOWER);
            ObjectSetInteger (0,«Sell»+i, OBJPROP_HIDDEN, true);
            ObjectSetString (0,«Sell»+i, OBJPROP_TOOLTIP, close [i-1]);
        }
    }
}
```

```
bars_calculated=calculated;
// -- return value of prev_calculated for next call
return (rates_total);
}
//+-----+
bool FillArrayFromPSARBuffer (double &sar_buffer [], // индикаторный буфер значений
Parabolic SAR
int ind_handle, // хэндл индикатора iSAR
int amount // количество копируемых значений
)
{
ResetLastError ();
if (CopyBuffer (ind_handle,0,0,amount, sar_buffer) <0)
{
return (false);
}
return (true);
}
//+-----+
bool FillArrayFromMABuffer (double &values [], // индикаторный буфер значений
Moving Average
int shift, // смещение
int ind_handle, // хэндл индикатора iMA
int amount // количество копируемых значений
)
{
ResetLastError ();
if (CopyBuffer (ind_handle,0, -shift, amount, values) <0)
{
return (false);
}
return (true);
}

void OnDeinit (const int reason) {
ObjectsDeleteAll (0, -1, -1);
}
```

Здесь добавлены счетчики сигналов на продажу и покупку countBuy, countSell, для того, чтобы сигнал звучал только при появлении первого сигнала.

Функция OnChartEvent

Функция OnChartEvent является функцией обратного вызова, которая вызывается при взаимодействии пользователя с графиком символа и событиях, связанных с графическими объектами графика символа.

```
void OnChartEvent (const int id, // идентификатор события
const long& lparam, // параметр события типа long
const double& dparam, // параметр события типа double
const string& sparam // параметр события типа string
);
```

В качестве примера использования функции OnChartEvent рассмотрим наш индикатор Impulse keeper и добавим в него функциональность, позволяющую посмотреть значения используемых индикаторов при клике на сигнале покупки или продажи индикатора.

Для этого добавим в код индикатора функцию OnChartEvent, обрабатывающую событие щелчка мыши на графическом объекте индикатора:

```
#property indicator_chart_window
#property indicator_buffers 4
double EMA34HBuffer [];
double EMA34LBuffer [];
double EMA125Buffer [];
double PSARBuffer [];
int EMA34HHandle;
int EMA34LHandle;
int EMA125Handle;
int PSARHandle;

int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit ()
{
// - - indicator buffers mapping
SetIndexBuffer (0,EMA34HBuffer, INDICATOR_CALCULATIONS);
SetIndexBuffer (1,EMA34LBuffer, INDICATOR_CALCULATIONS);
SetIndexBuffer (2,EMA125Buffer, INDICATOR_CALCULATIONS);
SetIndexBuffer (3,PSARBuffer, INDICATOR_CALCULATIONS);
EMA34HHandle=iMA (NULL,0,34,0,MODE_EMA, PRICE_HIGH);
EMA34LHandle=iMA (NULL,0,34,0,MODE_EMA, PRICE_LOW);
EMA125Handle=iMA (NULL,0,125,0,MODE_EMA, PRICE_CLOSE);
PSARHandle=iSAR (NULL,0,0.02, 0.2);
// - -
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 [])
{
// — —
int values_to_copy;
int start;
int calculated=BarsCalculated (EMA34HHandle);
if (calculated <=0)
{
return (0);
}
if (prev_calculated==0 || calculated!=bars_calculated)
{
start=rates_total-1;
if (calculated> rates_total) values_to_copy=rates_total;
else values_to_copy=calculated;
}
else
{
start=1;
values_to_copy=1;
}
if (!FillArrayFromMABuffer (EMA34HBuffer,0,EMA34HHandle, values_to_copy))
return (0); if (!FillArrayFromMABuffer (EMA34LBuffer,0,EMA34LHandle, values_to_copy))
return (0); if (!FillArrayFromMABuffer (EMA125Buffer,0,EMA125Handle, values_to_copy))
return (0);
if (!FillArrayFromPSARBuffer (PSARBuffer, PSARHandle, values_to_copy)) return (0);

ArraySetAsSeries (time, true);
ArraySetAsSeries (high, true);
ArraySetAsSeries (low, true);
ArraySetAsSeries (open, true);
ArraySetAsSeries (close, true);
ArraySetAsSeries (EMA34HBuffer, true);
ArraySetAsSeries (EMA34LBuffer, true);
ArraySetAsSeries (EMA125Buffer, true);
ArraySetAsSeries (PSARBuffer, true);

for (int i=start; i>=1;i - )
{if (close [i]> open [i] &&close [i]> EMA34HBuffer [i] &&close [i]> EMA34LBuffer
[i] &&low [i]> EMA125Buffer [i] &&low [i]> PSARBuffer [i] &&EMA125Buffer [i]
<EMA34LBuffer [i] &&EMA125Buffer [i] <EMA34HBuffer [i]) {
```

```

if (!ObjectCreate (0,«Buy»+i, OBJ_ARROW,0,time [i],high [i]))
{
return (false);
}
ObjectSetInteger (0,«Buy»+i, OBJPROP_COLOR, clrGreen);
ObjectSetInteger (0,«Buy»+i, OBJPROP_ARROWCODE,233);
ObjectSetInteger (0,«Buy»+i, OBJPROP_WIDTH,2);
ObjectSetInteger (0,«Buy»+i, OBJPROP_ANCHOR, ANCHOR_UPPER);
ObjectSetInteger (0,«Buy»+i, OBJPROP_HIDDEN, true);
ObjectSetString (0,«Buy»+i, OBJPROP_TOOLTIP, close [i]);
}
if (close [i] <open [i] &&close [i] <EMA34HBuffer [i] &&close [i] <EMA34LBuffer
[i] &&high [i] <EMA125Buffer [i] &&high [i] <PSARBuffer [i] &&EMA125Buffer [i]>
EMA34LBuffer [i] &&EMA125Buffer [i]> EMA34HBuffer [i]) {
if (!ObjectCreate (0,«Sell»+i, OBJ_ARROW,0,time [i],low [i]))
{
return (false);
}
ObjectSetInteger (0,«Sell»+i, OBJPROP_COLOR, clrRed);
ObjectSetInteger (0,«Sell»+i, OBJPROP_ARROWCODE,234);
ObjectSetInteger (0,«Sell»+i, OBJPROP_WIDTH,2);
ObjectSetInteger (0,«Sell»+i, OBJPROP_ANCHOR, ANCHOR_LOWER);
ObjectSetInteger (0,«Sell»+i, OBJPROP_HIDDEN, true);
ObjectSetString (0,«Sell»+i, OBJPROP_TOOLTIP, close [i]);
}
}
bars_calculated=calculated;
// - - return value of prev_calculated for next call
return (rates_total);
}
//+-----+
bool FillArrayFromPSARBuffer (double &sar_buffer [], // индикаторный буфер значений
Parabolic SAR
int ind_handle, // хэндл индикатора iSAR
int amount // количество копируемых значений
)
{
ResetLastError ();
if (CopyBuffer (ind_handle,0,0,amount, sar_buffer) <0)
{
return (false);
}
return (true);
}
//+-----+
bool FillArrayFromMABuffer (double &values [], // индикаторный буфер значений
Moving Average
int shift, // смещение

```

```
int ind_handle, // хэндл индикатора iMA
int amount // количество копируемых значений
)
{
ResetLastError ();
if (CopyBuffer (ind_handle,0, -shift, amount, values) <0)
{
return (false);
}
return (true);
}

void OnDeinit (const int reason) {
ObjectsDeleteAll (0, -1, -1);
}
//+-----+
//| ChartEvent function |
//+-----+
void OnChartEvent (const int id,
const long &lparam,
const double &dparam,
const string &sparam)
{
// —
if (id==CHARTEVENT_OBJECT_CLICK) {

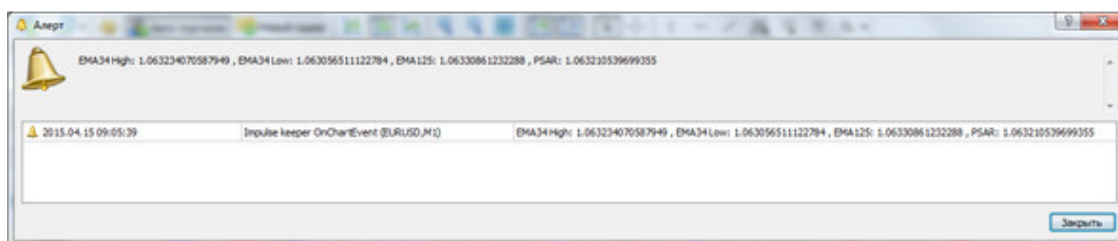
if (StringFind (sparam,«Sell», 0)!=-1) {
int pos=StringToInteger (StringSubstr (sparam,4));
Alert («EMA34 High:», EMA34HBuffer [pos],», EMA34 Low:», EMA34LBuffer [pos], ",
EMA125:», EMA125Buffer [pos], ", PSAR:», PSARBuffer [pos]);
}

if (StringFind (sparam,«Buy», 0)!=-1) {
int pos=StringToInteger (StringSubstr (sparam,3));
Alert («EMA34 High:», EMA34HBuffer [pos],», EMA34 Low:», EMA34LBuffer [pos], ",
EMA125:», EMA125Buffer [pos], ", PSAR:», PSARBuffer [pos]);
}
}
}
//+-----+
```

Здесь мы в функции OnCalculate с помощью функции ArraySetAsSeries изменили порядок доступа ко всем используемым массивам и здесь нам не нужно использовать i-1, так как в текущем тике мы начинаем цикл с индекса 1.

В функции OnChartEvent код сначала проверяет идентификатор события и если событие – это щелчок мыши на графическом объекте, код переходит к проверке, является ли этот объект графическим объектом индикатора.

Дальше код выделяет из имени объекта его порядковый номер, который соответствует индексу бара, и выводит значения буферов используемых индикаторов в диалоговое окно Alert отображения информации пользователю.



Параллельно информация выводится в окно Эксперты терминала.

Здесь было бы удобно вывести информацию в диалоговое окно `MessageBox`, которое позволяет взаимодействовать с пользователем, но его нельзя вызывать из пользовательских индикаторов, так как индикаторы выполняются в интерфейсном потоке и не должны его тормозить.

Объектно-ориентированный подход

В объектно-ориентированных языках все – это классы. В том числе и точка входа в приложение является классом, который содержит специфический метод – точку входа в приложение, или который расширяет специфический класс фреймворка или платформы.

С языком MQL5 это немного не так. Создание программ на языке MQL5 связано с использованием набора функций обратного вызова, которые вызываются клиентским терминалом при наступлении тех или иных событий. И код MQL5-приложения не является классом, а состоит из набора функций обратного вызова и вспомогательного пользовательского кода. Так вот в части именно вспомогательного пользовательского кода разработчик и волен использовать либо процедурное программирование, либо объектно-ориентированное программирование.

В случае выбора процедурного программирования вспомогательный пользовательский код представляет собой набор пользовательских функций, при выборе объектно-ориентированного программирования вспомогательный пользовательский код представляет собой набор пользовательских классов, которые могут использовать стандартную MQL5-библиотеку классов.

Напомним базовые понятия объектно-ориентированного программирования.

Инкапсуляция – это когда код представлен классами, которые предоставляют открытые методы для доступа и изменения данных, таким образом защищая данные.

Расширяемость типов – это возможность добавлять пользовательские типы данных, что как раз основано на использовании классов, так как каждый новый пользовательский класс представляет новый тип данных.

Наследование – это возможность создавать новые классы на основе уже существующих классов, таким образом, повторно используя уже существующий проверенный и протестированный код. При этом в MQL5 нет множественного наследования.

Полиморфизм – это возможность иметь всем классам одной и той же иерархии наследования метод с одним именем, но разной реализацией.

Перегрузка – это создание методов класса, имеющих одно имя, но предназначенных для работы с разными типами данных, таким образом класс делается универсальным для разных типов данных.

В качестве примера использования объектно-ориентированного подхода рассмотрим создание нашего пользовательского индикатора Impulse keeper с применением классов.

В данном случае, использование класса CIndicator и его наследников CiMA и CiSAR, обеспечивающих доступ к индикаторам MA и PSAR, позволяет вообще обойтись без буферов индикатора Impulse keeper. Так как они были нужны нам для копирования буферов индикаторов MA и PSAR, а классы CiMA и CiSAR предоставляют напрямую доступ к своим буферам.

Для использования класса CIndicator и его потомков, в код необходимо включить файл Trend.mqh:

```
#include <Indicators\Trend.mqh>
```

Далее в коде индикатора Impulse keeper объявим экземпляры классов CiMA и CiSAR:

```
#property indicator_chart_window
```

```
CiMA MA34H;  
CiMA MA34L;  
CiMA MA125;  
CiSAR SAR;
```

```
int bars_calculated=0;
```

В функции OnInit () создадим индикаторы:

```
int OnInit ()
{
MA34H.Create (_Symbol, PERIOD_CURRENT,34,0,MODE_EMA, PRICE_HIGH);
MA34L.Create (_Symbol, PERIOD_CURRENT,34,0,MODE_EMA, PRICE_LOW);
MA125.Create (_Symbol, PERIOD_CURRENT,125,0,MODE_EMA, PRICE_CLOSE);
SAR.Create (_Symbol, PERIOD_CURRENT,0.02, 0.2);
// — —
return (INIT_SUCCEEDED);
}
```

В функции OnCalculate после вычисления начальной позиции расчета индикатора установим размеры буферов индикаторов CiMA и CiSAR:

```
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 start;
int calculated=MA34H.BarsCalculated ();
if (calculated <=0)
{
return (0);
}
if (prev_calculated==0 || calculated!=bars_calculated)
{
start=rates_total-1;
}
else
{
start=1;
}
ArraySetAsSeries (time, true);
ArraySetAsSeries (high, true);
ArraySetAsSeries (low, true);
ArraySetAsSeries (open, true);
ArraySetAsSeries (close, true);

//Print (MA34H. BufferSize ());

MA34H. BufferResize (rates_total);
```

```
MA34L.BufferResize (rates_total);
MA125.BufferResize (rates_total);
SAR.BufferResize (rates_total);
```

Если это не сделать, размеры буферов используемых индикаторов будут по умолчанию иметь величину 100, и наш индикатор будет рассчитываться только до 100 бара.

Далее обновим данные используемых индикаторов и рассчитаем и отрисуем наш индикатор:

```
MA34H.Refresh ();
MA34L.Refresh ();
MA125.Refresh ();
SAR.Refresh ();
for (int i=start; i>=1;i--)
{
if(close[i]>open[i]&&close[i]>MA34H.Main(i)&&close[i]>MA34L.Main(i)&&low[i]>MA125.Ma
if (!ObjectCreate (0,«Buy»+i, OBJ_ARROW,0,time [i],high [i]))
{
return (false);
}
ObjectSetInteger (0,«Buy»+i, OBJPROP_COLOR, clrGreen);
ObjectSetInteger (0,«Buy»+i, OBJPROP_ARROWCODE,233);
ObjectSetInteger (0,«Buy»+i, OBJPROP_WIDTH,2);
ObjectSetInteger (0,«Buy»+i, OBJPROP_ANCHOR, ANCHOR_UPPER);
ObjectSetInteger (0,«Buy»+i, OBJPROP_HIDDEN, true);
ObjectSetString (0,«Buy»+i, OBJPROP_TOOLTIP, close [i]);
}
if(close[i]<open[i]&&close[i]<MA34H.Main(i)&&close[i]<MA34L.Main(i)&&high[i]<MA125.Ma
if (!ObjectCreate (0,«Sell»+i, OBJ_ARROW,0,time [i],low [i]))
{
return (false);
}
ObjectSetInteger (0,«Sell»+i, OBJPROP_COLOR, clrRed);
ObjectSetInteger (0,«Sell»+i, OBJPROP_ARROWCODE,234);
ObjectSetInteger (0,«Sell»+i, OBJPROP_WIDTH,2);
ObjectSetInteger (0,«Sell»+i, OBJPROP_ANCHOR, ANCHOR_LOWER);
ObjectSetInteger (0,«Sell»+i, OBJPROP_HIDDEN, true);
ObjectSetString (0,«Sell»+i, OBJPROP_TOOLTIP, close [i]);
}
}
ChartRedraw (0);
bars_calculated=calculated;
// -- return value of prev_calculated for next call
return (rates_total);
}
//+-----+
void OnDeinit (const int reason) {
ObjectsDeleteAll (0, -1, -1);
}
}
```

Чтобы быть последовательными в объектно-ориентированном подходе, весь код по расчету и отрисовке нашего индикатора можно выделить в отдельный пользовательский

класс. Благо мастер редактора MQL5 предоставляет возможность создания каркаса пользовательского класса.

```
#include <Indicators\Trend.mqh>
//+-----+
//|
//+-----+
class IKSIGNAL
{
private:
int _start;
datetime _time [];
double _open [];
double _high [];
double _low [];
double _close [];

public:
IKSignal (
int start,
const datetime &time [],
const double &open [],
const double &high [],
const double &low [],
const double &close []
);
bool draw (CiMA &MA34H, CiMA &MA34L, CiMA &MA125, CiSAR &SAR);
~IKSignal ();
};
//+-----+
//|
//+-----+
IKSignal::IKSignal (int start,
const datetime &time [],
const double &open [],
const double &high [],
const double &low [],
const double &close []
)
{
_start=start;
if (ArraySize (time)> 0)
{
ArrayResize (_time, ArraySize (time));
ArrayCopy (_time, time);
}
if (ArraySize (open)> 0)
{
ArrayResize (_open, ArraySize (open));
ArrayCopy (_open, open);
}
```

```

    }
    if (ArraySize (high)> 0)
    {
        ArrayResize (_high, ArraySize (high));
        ArrayCopy (_high, high);
    }
    if (ArraySize (low)> 0)
    {
        ArrayResize (_low, ArraySize (low));
        ArrayCopy (_low, low);
    }
    if (ArraySize (close)> 0)
    {
        ArrayResize (_close, ArraySize (close));
        ArrayCopy (_close, close);
    }
    ArraySetAsSeries (_time, true);
    ArraySetAsSeries (_high, true);
    ArraySetAsSeries (_low, true);
    ArraySetAsSeries (_open, true);
    ArraySetAsSeries (_close, true);
}
//+-----+
//|
//+-----+
IKSignal::~IKSignal ()
{
}
//+-----+
bool IKSignal::draw (CiMA &MA34H, CiMA &MA34L, CiMA &MA125, CiSAR
&SAR) {
    for (int i=_start; i>=1;i--)
    {if(_close[i]>_open[i]&&_close[i]>MA34H.Main(i)&&_close[i]>MA34L.Main(i)&&_low[i]>MA1
    if (!ObjectCreate (0,«Buy»+i, OBJ_ARROW,0,_time [i],_high [i]))
    {
        return (false);
    }
    ObjectSetInteger (0,«Buy»+i, OBJPROP_COLOR, clrGreen);
    ObjectSetInteger (0,«Buy»+i, OBJPROP_ARROWCODE,233);
    ObjectSetInteger (0,«Buy»+i, OBJPROP_WIDTH,2);
    ObjectSetInteger (0,«Buy»+i, OBJPROP_ANCHOR, ANCHOR_UPPER);
    ObjectSetInteger (0,«Buy»+i, OBJPROP_HIDDEN, true);
    ObjectSetString (0,«Buy»+i, OBJPROP_TOOLTIP,_close [i]);
}
    if(_close[i]<_open[i]&&_close[i]<MA34H.Main(i)&&_close[i]<MA34L.Main(i)&&_high[i]<MA1
    if (!ObjectCreate (0,«Sell»+i, OBJ_ARROW,0,_time [i],_low [i]))
    {
        return (false);
    }
}

```

```
ObjectSetInteger (0,«Sell»+i, OBJPROP_COLOR, clrRed);
ObjectSetInteger (0,«Sell»+i, OBJPROP_ARROWCODE,234);
ObjectSetInteger (0,«Sell»+i, OBJPROP_WIDTH,2);
ObjectSetInteger (0,«Sell»+i, OBJPROP_ANCHOR, ANCHOR_LOWER);
ObjectSetInteger (0,«Sell»+i, OBJPROP_HIDDEN, true);
ObjectSetString (0,«Sell»+i, OBJPROP_TOOLTIP,_close [i]);
}
}
ChartRedraw (0);
return (true);
}
```

Здесь в классе `IKSignal` мы объявляем закрытые поля класса, представляющие начальную позицию расчета индикатора и ценовую историю.

В конструкторе класса мы копируем его параметры в поля класса и меняем порядок доступа к полям-массивам.

Также в классе объявляется открытая функция `draw`, в которой фактически и будет производиться расчет и отрисовка индикатора.

В качестве параметров этой функции выступают экземпляры классов `CiMA` и `CiSAR`. Тут мы просто переносим код из функции `OnCalculate`.

Теперь в коде основного файла нам не нужно включать файл `Trend.mqh`, так как мы уже сделали это в коде класса `IKSignal`, вместо этого нам нужно включить файл класса `IKSignal`.

Поместим файл класса `IKSignal` в каталог `Include` и включим его в основной файл индикатора:

```
#include <IKSignal.mqh>
Теперь функция OnCalculate примет следующий вид:
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 start;
int calculated=MA34H.BarsCalculated ();
if (calculated <=0)
{
return (0);
}
if (prev_calculated==0 || calculated!=bars_calculated)
{
start=rates_total-1;
}
else
{
}
```

```
start=1;
}
IKSignal iks (start, time, open, high, low, close)
//Print (MA34H. BufferSize ());

MA34H. BufferResize (rates_total);
MA34L. BufferResize (rates_total);
MA125.BufferResize (rates_total);
SAR. BufferResize (rates_total);

MA34H.Refresh ();
MA34L.Refresh ();
MA125.Refresh ();
SAR.Refresh ();

if (!iks. draw (MA34H, MA34L, MA125, SAR)) {
return (false);
}
bars_calculated=calculated;
// – - return value of prev_calculated for next call
return (rates_total);
}
```

Здесь мы создаем экземпляр класса IKSignal с указанными в правильном порядке параметрами и применяем к нему функцию draw.

Как видно, код основного файла индикатора значительно упрощается.

Общая структура советника

Советник или эксперт это MQL5-программа, способная автоматически выставить и закрывать ордера на покупку и продажу финансового инструмента, таким образом, осуществляя автоматическую торговлю в клиентском терминале.

Подобно индикаторам, код экспертов основан на функциях обратного вызова, вызываемых клиентским терминалом при наступлении определенных событий.

Для эксперта это такие функции как OnInit (), OnDeinit, OnTick (), OnTimer (), OnTrade (), OnTradeTransaction, OnTester (), OnTesterInit (), OnTesterPass (), OnTesterDeinit (), OnBookEvent (), OnChartEvent ().

Впрочем, для организации автоматической торговли достаточно двух функций OnInit () и OnTick ().

В отличие от индикаторов, для экспертов особо никакие свойства не объявляются, за исключением link, copyright, version и description, и если эксперт попутно с торговлей не рисует индикатор. Поэтому перед функциями обратного вызова, в эксперте, объявляются входные параметры, хэндлы используемых технических индикаторов, глобальные переменные и константы.

Здесь правда есть один параметр, которого нет в индикаторе, но который присутствует для эксперта.

Это магическое число или идентификатор эксперта.

С помощью магического числа идентифицируются торговые ордера, выставаемые экспертом. Это дает возможность создания взаимосвязанной системы из нескольких работающих экспертов.

В функции OnInit () эксперта производится инициализация хэндлов используемых технических индикаторов и переменных эксперта.

В функции OnDeinit эксперта, как правило, производится удаление глобальных переменных клиентского терминала, созданных экспертом во время тестирования, оптимизации или отладки, а также освобождение расчетной части используемого индикатора и удаление индикатора из графика по окончании тестирования эксперта с помощью функции IndicatorRelease ().

Глобальные переменные клиентского терминала отличаются от глобальных переменных MQL5-приложения. Глобальные переменные клиентского терминала могут быть созданы MQL5-приложением с помощью функции GlobalVariableSet, но при этом они становятся доступными для других MQL5-приложений клиентского терминала, в отличие от глобальных переменных MQL5-приложения. Таким образом, глобальные переменные клиентского терминала это средство коммуникации между разными MQL5-приложениями. Как правило, глобальные переменные клиентского терминала создаются экспертами для проверки на истечение временного лимита для предыдущей сделки.

Сами по себе глобальные переменные существуют в клиентском терминале 4 недели с момента последнего обращения, после этого автоматически уничтожаются. Обращением к глобальной переменной считается не только установка нового значения, но и чтение значения глобальной переменной.

При тестировании эксперта глобальные переменные клиентского терминала эмулируются, и они никак не связаны с настоящими глобальными переменными терминала. Все операции с глобальными переменными терминала при тестировании эксперта производятся вне клиентского терминала в агенте тестирования.

Принудительно глобальные переменные клиентского терминала можно уничтожить с помощью вызова функции GlobalVariableDel или GlobalVariablesDeleteAll.

В функции OnTick () эксперта сначала производится проверка возможности торговли на данном счете, достаточности средств на счете, достаточности загруженной ценовой истории для расчетов торговой стратегии. Затем устанавливаются временные фильтры для совершения торговых операций, проверяется наличие открытых позиций, производятся вычисления торговой стратегии и на основании ее сигналов открываются или закрываются позиции или выставляются отложенные ордера.

Функция OnTimer () позволяет создать альтернативную модель эксперта, который будет производить вычисления торговой стратегии не при наступлении нового тика в функции OnTick (), а через определенные функцией EventSetTimer промежутки времени. При этом в функции OnDeinit эксперта нужно удалить таймер с помощью вызова функции EventKillTimer.

Функция OnTrade () позволяет обработать завершение торговой операции. Торговая операция это не только открытие или закрытие позиции, это также установка, модификация или удаление отложенного ордера, отмена отложенного ордера при нехватке средств либо по истечении срока действия, срабатывание отложенного ордера, модификация открытой позиции. Например, функцию OnTrade () можно использовать для временного ограничения торговли при срабатывании Stop Loss.

Изменение состояния торгового счета происходит в результате серии транзакций, например, создание ордера, исполнение ордера, удаление ордера из списка открытых, добавление в историю ордеров, добавление сделки в историю, создание новой позиции. Каждая такая транзакция сопровождается вызовом функции OnTradeTransaction, в которой можно обрабатывать выполнение транзакции. В частности, в функции OnTradeTransaction можно обрабатывать результат исполнения торгового запроса на сервере, отправленного функцией OrderSendAsync ().

При тестировании эксперта в режиме тестирования с использованием генетического алгоритма производится подбор наилучшей комбинации входных параметров эксперта по критерию оптимизации. Изначально тестер предлагает набор предопределенных критериев оптимизации, таких как максимальный баланс счета, баланс + максимальная прибыльность, баланс + минимальная просадка и другие.

Функция OnTester () позволяет определить свой критерий оптимизации, так как при оптимизации тестер всегда ищет локальный максимум возвращаемого значения функции OnTester (), которая автоматически вызывается после окончания очередного прохода тестирования эксперта на заданном интервале дат. Для использования функции OnTester () в тестере выбирается критерий оптимизации Custom max.

Функции OnTesterInit (), OnTesterPass (), OnTesterDeinit () позволяют организовать динамическую обработку результатов оптимизации параметров эксперта в тестере при каждом проходе оптимизации.

Функция OnBookEvent () позволяет разработать советник или индикатор, использующий торговую стратегию, которая основана на стакане цен, если конечно брокер предоставляет такую возможность.

Функция OnChartEvent (), так же как и функция OnTimer (), позволяет создать альтернативную модель эксперта, который будет производить вычисления торговой стратегии не при наступлении нового тика в функции OnTick (), а при получении событий от индикаторов, прикрепленных к графику символа.

Функция OnTick ()

Как уже было сказано, в функции OnTick (), код, как правило, перед вычислениями торговой стратегии начинается с различного рода проверок, хотя некоторые проверки можно выполнить и в функции OnInit ().

Информацию о счете клиента можно получить с помощью функций AccountInfoDouble, AccountInfoInteger и AccountInfoString.

В качестве аргумента этих функций указывается идентификатор свойства, значение которого нужно получить.

Для функции AccountInfoInteger это следующие свойства:

ACCOUNT_LOGIN – функция возвращает номер счета.

ACCOUNT_TRADE_MODE – функция возвращает тип торгового счета. Функция возвращает 0 для демонстрационного торгового счета, 1 для конкурсного торгового счета, 2 для реального торгового счета.

ACCOUNT_LEVERAGE – возвращает размер кредитного плеча счета, например, для плеча 1:100, функция вернет 100.

ACCOUNT_LIMIT_ORDERS – функция возвращает максимальное разрешенное количество отложенных ордеров. Такое ограничение устанавливается брокером, и если ограничений нет, функция возвращает 0.

ACCOUNT_MARGIN_SO_MODE – в чем задается минимально допустимый уровень залоговых средств, в процентах или в деньгах. Минимально допустимый уровень залоговых средств это уровень залоговых средств, при котором требуется или пополнение счета, или уровень залоговых средств, при достижении которого происходит принудительное закрытие самой убыточной позиции. Минимально допустимый уровень залоговых средств устанавливается брокером и функция возвращает 0, если уровень задается в процентах, и возвращает 1, если уровень задается в деньгах.

ACCOUNT_TRADE_ALLOWED – функция возвращает 0, если для счета запрещена торговля в случае подключения к счету в режиме инвестора, отсутствия соединения к серверу, запрета торговли на стороне сервера, если счет отправлен в архив. Функция возвращает 1, если торговля на счете разрешена.

ACCOUNT_TRADE_EXPERT – функция возвращает 0, если брокер запрещает автоматическую торговлю, и возвращает 1, если автоматическая торговля разрешена.

Свойство ACCOUNT_LOGIN может быть использовано для защиты эксперта с помощью его привязки к конкретному счету.

Для этого можно объявить константу, представляющую валидный номер счета и в функции OnInit () сравнить ее с текущим счетом:

```
const long _ACCOUNT=50009917;
```

```
if (AccountInfoInteger (ACCOUNT_LOGIN)!=_ACCOUNT) {  
    Print («Не соответствует номер счета»);  
    return (0);  
}
```

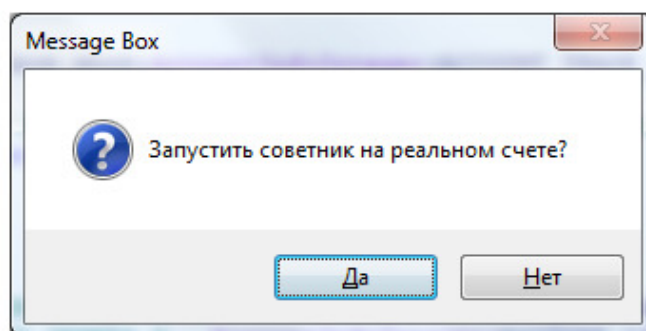
Значение свойства ACCOUNT_TRADE_MODE можно вывести в виде перечисления, для этого возвращаемое функцией значение нужно привести к перечислению, а затем конвертировать в строку:

```
Print (AccountInfoInteger (ACCOUNT_TRADE_MODE), EnumToString  
((ENUM_ACCOUNT_TRADE_MODE) AccountInfoInteger (ACCOUNT_TRADE_MODE)));
```

Свойство ACCOUNT_TRADE_MODE можно использовать для проверки в функции OnInit () запуска эксперта на реальном счете:

```
if ((ENUM_ACCOUNT_TRADE_MODE) AccountInfoInteger  
(ACCOUNT_TRADE_MODE) == ACCOUNT_TRADE_MODE_REAL) {  
    int mb=MessageBox («Запустить советник на реальном счете?», «Message Box»,  
    MB_YESNO|MB_ICONQUESTION);  
    if (mb==IDNO) return (0);  
}
```

При этом отобразится диалоговое окно, которое при выборе кнопки Да позволит дальнейшее выполнение кода:



Свойство ACCOUNT_LIMIT_ORDERS может быть использовано для проверки и установки максимального количества отложенных ордеров:

```
bool IsNewOrderAllowed (int _max_orders)  
{  
    int orders=OrdersTotal ();  
    int max_allowed_orders= (int) AccountInfoInteger (ACCOUNT_LIMIT_ORDERS);  
    if (max_allowed_orders!=0&&_max_orders> max_allowed_orders) {  
        _max_orders=max_allowed_orders;  
    }  
    return (orders < _max_orders);  
}
```

Объявим входной параметр:

```
input int max_orders=1; //максимальное количество ордеров
```

И вызовем определенную нами функцию:

```
Print (IsNewOrderAllowed (max_orders));
```

Проверку свойств ACCOUNT_TRADE_ALLOWED и ACCOUNT_TRADE_EXPERT можно организовать в функции OnInit ():

```
// -----  
if (!TerminalInfoInteger (TERMINAL_CONNECTED)) {  
    Alert («No connection to the trade server»);  
    return (0);  
} else {  
    if (!AccountInfoInteger (ACCOUNT_TRADE_ALLOWED)) {  
        Alert («Trade for this account is prohibited»);  
        return (0);  
    }
```

```
}  
}  
if (!AccountInfoInteger (ACCOUNT_TRADE_EXPERT)) {  
Alert («Trade with the help of experts for the account is prohibited»);  
return (0);  
}  
// —————
```

Дополнительно отдельно проверку соединения с сервером можно сделать в функции OnTick ():

```
if (!TerminalInfoInteger (TERMINAL_CONNECTED)) {  
Alert («No connection to the trade server»);  
return;  
}
```

Для функции AccountInfoDouble определены следующие свойства:

ACCOUNT_BALANCE – баланс счета. Соответствует значению **Баланс** вкладке **Торговля** клиентского терминала.

```
Print (AccountInfoDouble (ACCOUNT_BALANCE));
```

ACCOUNT_CREDIT – размер предоставленного кредита. Типичная ситуация, когда это значение равно 0.

ACCOUNT_PROFIT – размер текущей прибыли на счете. Соответствует столбцу **Прибыль** во вкладке **Торговля** клиентского терминала.

```
Print (AccountInfoDouble (ACCOUNT_PROFIT));
```

ACCOUNT_EQUITY – значение собственных средств на счете. Соответствует значению **Средства** вкладке **Торговля** клиентского терминала.

```
Print (AccountInfoDouble (ACCOUNT_EQUITY));
```

ACCOUNT_MARGIN – размер зарезервированных залоговых средств на счете. Соответствует значению **Маржа** вкладке **Торговля** клиентского терминала. Если открытых позиций нет, это значение равно 0.

```
Print (AccountInfoDouble (ACCOUNT_MARGIN));
```

ACCOUNT_MARGIN_FREE – размер свободных средств на счете, доступных для открытия позиций. Соответствует значению **Свободная маржа** вкладке **Торговля** клиентского терминала.

```
Print (AccountInfoDouble (ACCOUNT_MARGIN_FREE));
```

ACCOUNT_MARGIN_LEVEL – уровень залоговых средств на счете в процентах. Соответствует значению **Уровень маржи** вкладке **Торговля** клиентского терминала. Рассчитывается как **Средства/Маржа*100%**. Если открытых позиций нет, это значение равно 0.

```
Print (AccountInfoDouble (ACCOUNT_MARGIN_LEVEL));
```

ACCOUNT_MARGIN_SO_CALL – уровень залоговых средств, при котором требуется пополнение счета (Margin Call). В зависимости от установленного ACCOUNT_MARGIN_SO_MODE выражается в процентах либо в валюте депозита. Margin Call это скорее информационный сигнал для трейдера, что его счет близок к закрытию, и не сопровождается действиями брокера. Последствия наступают в случае возникновения Stop Out. Например, при ACCOUNT_MARGIN_SO_CALL = 50%, событие Margin Call наступит, когда размер средств на счете станет как половина от маржи.

```
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger  
(ACCOUNT_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_PERCENT)
```

```
Print (AccountInfoDouble (ACCOUNT_MARGIN_SO_CALL),» %»);
```

```
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger  
(ACCOUNT_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_MONEY)
```

```
Print (AccountInfoDouble (ACCOUNT_MARGIN_SO_CALL),» USD»);
```

ACCOUNT_MARGIN_SO_SO – уровень залоговых средств, при достижении которого происходит принудительное закрытие самой убыточной позиции (Stop Out). В зависимости от установленного ACCOUNT_MARGIN_SO_MODE выражается в процентах либо в валюте депозита. Например, при ACCOUNT_MARGIN_SO_SO = 10%, событие Stop Out наступит, когда размер средств на счете будет 10% от маржи, при этом открытые позиции начнут принудительно закрываться брокером.

```
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger  
(ACCOUNT_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_PERCENT)
```

```
Print (AccountInfoDouble (ACCOUNT_MARGIN_SO_SO),» %»);
```

```
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger  
(ACCOUNT_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_MONEY)
```

```
Print (AccountInfoDouble (ACCOUNT_MARGIN_SO_SO),» USD»);
```

ACCOUNT_MARGIN_INITIAL – размер средств, зарезервированных на счете, для обеспечения гарантийной суммы по всем отложенным ордерам. Как правило, эта величина равна 0.

ACCOUNT_MARGIN_MAINTENANCE – размер средств, зарезервированных на счете, для обеспечения минимальной суммы по всем открытым позициям. Как правило, эта величина равна 0.

ACCOUNT_ASSETS – текущий размер активов на счете. Как правило, эта величина равна 0.

ACCOUNT_LIABILITIES – текущий размер обязательств на счете. Как правило, эта величина равна 0.

ACCOUNT_COMMISSION_BLOCKED – текущая сумма заблокированных комиссий по счету. Как правило, эта величина равна 0.

С помощью свойств функции AccountInfoDouble можно организовать различного рода проверки в функции OnTick () эксперта.

Наступление события Margin Call:

```
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger  
(ACCOUNT_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_PERCENT) {
```

```
if (AccountInfoDouble (ACCOUNT_MARGIN_LEVEL) != 0 && AccountInfoDouble  
(ACCOUNT_MARGIN_LEVEL) <= AccountInfoDouble (ACCOUNT_MARGIN_SO_CALL))
```

```
Alert («Margin Call!!!»);
```

```
}
```

```
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger  
(ACCOUNT_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_MONEY) {
```

```
if (AccountInfoDouble (ACCOUNT_EQUITY) <= AccountInfoDouble  
(ACCOUNT_MARGIN_SO_CALL))
```

```
Alert («Margin Call!!!»);
```

```
}
```

Наступление события Stop Out:

```
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger  
(ACCOUNT_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_PERCENT) {
```

```
if (AccountInfoDouble (ACCOUNT_MARGIN_LEVEL) != 0 && AccountInfoDouble  
(ACCOUNT_MARGIN_LEVEL) <= AccountInfoDouble (ACCOUNT_MARGIN_SO_SO)) {
```

```
Alert («Stop Out!!!»);
```

```
return;
```

```
}
```

```
}
```

```
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger
(Account_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_MONEY) {
    if (AccountInfoDouble (ACCOUNT_EQUITY) <= AccountInfoDouble
(Account_MARGIN_SO_SO)) {
        Alert («Stop Out!!!»);
        return;
    }
}
Проверка размера свободных средств на счете, доступных для открытия позиции:
double margin;
MqlTick last_tick;
ResetLastError ();
if (SymbolInfoTick (Symbol (),last_tick))
{if(OrderCalcMargin(ORDER_TYPE_BUY,Symbol(),Lot,last_tick.ask, margin))
{
    if (margin> AccountInfoDouble (ACCOUNT_MARGIN_FREE)) {
        Alert («Not enough money in the account!»);
        return;
    }
}
}
else
{
    Print (GetLastError ());
}
```

Здесь MqlTick это стандартная структура для хранения цен, которая заполняется значениями с помощью функции SymbolInfoTick.

Вызов функции ResetLastError () производится для обнуления ошибки перед вызовом функции, после которой проверяется возникновение ошибки.

Функция OrderCalcMargin вычисляет размер средств, необходимых для открытия позиции. И если размер свободных средств на счете (ACCOUNT_MARGIN_FREE) меньше, чем размер средств, необходимых для открытия позиции, денег на счете недостаточно и торговля невозможна.

Для функции AccountInfoString определены такие свойства как имя клиента ACCOUNT_NAME, имя торгового сервера ACCOUNT_SERVER, валюта депозита ACCOUNT_CURRENCY, имя компании, обслуживающей счет ACCOUNT_COMPANY.

С помощью свойства ACCOUNT_NAME, также как и с помощью свойства ACCOUNT_LOGIN, можно защитить советник:

```
if (AccountInfoString (ACCOUNT_NAME) != _name) {
    Print («Не соответствует имя пользователя»);
    return (0);
}
```

Информацию о клиентском терминале можно получить с помощью функций TerminalInfoInteger () и TerminalInfoString (). В качестве аргумента эти функции также принимают свойства.

Мы уже видели проверку подключения терминала к серверу с помощью свойства TERMINAL_CONNECTED.

Свойство TERMINAL_DLLS_ALLOWED позволяет выяснить, есть ли разрешение на использование DLL:

```
Print ((bool) TerminalInfoInteger (TERMINAL_DLLS_ALLOWED));
```

Файлы DLL это еще один способ создания повторно используемых библиотек – модулей кода для MQL5-программ.

DLL-библиотеки находятся в папке MQL5\Libraries торгового терминала и включаются в код MQL5-программы с помощью команды:

```
#import «dll_lib.dll»
```

При этом разрешение на использование DLL-библиотек устанавливается во вкладке Советники настроек клиентского терминала.

DLL-библиотеки могут также применяться для защиты эксперта с помощью переноса основного кода торговой стратегии в DLL-файл.

Свойство TERMINAL_TRADE_ALLOWED показывает, включена ли кнопка авто-торговли в клиентском терминале. Для проверки этого свойства в функцию OnTick () можно включить код:

```
if (!TerminalInfoInteger (TERMINAL_TRADE_ALLOWED))
```

```
Alert («Разрешение на автоматическую торговлю выключено!»);
```

Однако разрешение на торговлю с помощью эксперта может быть отключено в общих свойствах самого эксперта. Для проверки этого условия можно использовать свойство MQL_TRADE_ALLOWED функции MQLInfoInteger:

```
if (!MQLInfoInteger (MQL_TRADE_ALLOWED))
```

```
Alert («Автоматическая торговля запрещена в свойствах эксперта», __FILE__);
```

С помощью свойства SYMBOL_SPREAD функции SymbolInfoInteger можно осуществить контроль над спредом брокера:

```
double _spread=SymbolInfoInteger (Symbol (),SYMBOL_SPREAD) *MathPow (10, -  
SymbolInfoInteger (Symbol (),SYMBOL_DIGITS)) /MathPow (10, -4);
```

```
if (_spread> spreadLevel) {
```

```
Alert («Слишком большой спред!»);
```

```
return;
```

```
}
```

Здесь с помощью свойства SYMBOL_DIGITS выясняем, сколько знаков после запятой в цене и вычисляем спред в пунктах. Затем сравниваем его с пороговым значением, и если текущий спред больше порогового значения, торговлю не осуществляем.

С помощью свойства SYMBOL_TRADE_MODE функции SymbolInfoInteger можно проверить ограничения на торговые операции по символу, установленные брокером:

```
if ((ENUM_SYMBOL_TRADE_MODE) SymbolInfoInteger (Symbol  
( ),SYMBOL_TRADE_MODE)!=SYMBOL_TRADE_MODE_FULL) {
```

```
Alert («Установлены ограничения на торговые операции»);
```

```
return;
```

```
}
```

В функции OnTick () можно ограничить работу эксперта по времени.

Если вы хотите, чтобы эксперт работал каждый день в заданный интервал времени, определите начальные и конечные час и минуты временного интервала и сравните их с текущим временем:

```
input int startHour=0; //start hour
```

```
input int startMin=0; //start minute
```

```
input int stopHour=23; // stop hour
```

```
input int stopMin=0; // stop minute
```

```
long _seconds=TimeLocal () %86400;
```

```
long startSec=3600*startHour+60*startMin;
```

```
long stopSec=3600*stopHour+60*stopMin;  
if (_seconds <startSec || _seconds> stopSec) return;
```

Здесь берется локальное время, если вы хотите сравнивать с серверным временем, используйте функцию TimeCurrent ().

Если вы хотите, чтобы эксперт просто отработал в заданный интервал времени, определите начальную и конечную даты временного интервала и сравните их с текущим временем:

```
input datetime startSession=D'2015.05.05»;  
input datetime stopSession=D'2015.05.06»;
```

```
datetime _session=TimeLocal ();  
if (_session <startSession || _session> stopSession) return;
```

В функции OnTick () эксперта также было бы неплохо проверить, достаточно ли баров в истории для расчета советника. Сделать это можно двумя способами – с помощью функции Bars и с помощью свойства SERIES_BARS_COUNT функции SeriesInfoInteger:

```
if (Bars (Symbol (), 0) <100)  
{  
Alert («In the chart little bars, Expert will not work!!»);  
return;  
}  
Или  
if (SeriesInfoInteger (Symbol (),0,SERIES_BARS_COUNT) <100)  
{  
Alert («In the chart little bars, Expert will not work!!»);  
return;  
}
```

Ограничить вычисления советника в функции OnTick () по появлению нового бара на графике также можно двумя способами, с помощью свойства SERIES_LASTBAR_DATE функции SeriesInfoInteger или с помощью функции CopyTime:

```
static datetime last_time;  
datetime last_bar_time= (datetime) SeriesInfoInteger (Symbol (),Period (),  
SERIES_LASTBAR_DATE);  
if (last_time!=last_bar_time)  
{  
last_time=last_bar_time;  
} else {  
return;  
}  
Или  
static datetime Old_Time;  
datetime New_Time [1];  
int copied=CopyTime (Symbol (),Period (),0,1,New_Time);  
ResetLastError ();  
if (copied> 0)  
{  
if (Old_Time!=New_Time [0])  
{  
Old_Time=New_Time [0];  
} else {
```

```
return;  
}  
}  
else  
{  
Print (GetLastError ());  
return;  
}
```

Если вы, предположим, хотите, после того как поймали StopLoss, прекратить на сегодня торговлю советником, вам нужно правильно обработать это StopLoss событие.

Как известно, функция OnTrade () вызывается при открытии или закрытии позиции, установке, модификации или удалении отложенного ордера, отмене отложенного ордера при нехватке средств либо по истечении срока действия, срабатывании отложенного ордера, модификации открытой позиции. Поэтому в функции OnTrade () нужно выделить только события совершения сделок, а затем из сделок выделить событие закрытия позиции по StopLoss:

```
void OnTrade ()  
{  
static int _deals;  
ulong _ticket=0;  
  
if (HistorySelect (0,TimeCurrent ()))  
{  
int i=HistoryDealsTotal () -1;  
  
if (_deals!=i) {  
_deals=i;  
} else {return;}  
  
if ((_ticket=HistoryDealGetTicket (i))> 0)  
{  
string _comment=HistoryDealGetString (_ticket, DEAL_COMMENT);  
  
if (StringFind (_comment,«sl», 0)!=-1) {  
flagStopLoss=true;  
}  
  
}  
}  
}
```

Здесь функция HistorySelect запрашивает историю сделок и ордеров за все время, затем с помощью функции HistoryDealsTotal мы получаем индекс последней сделки и сравниваем его со статической переменной, хранящей индекс предыдущей сделки. Таким образом, мы выделяем только события совершения сделок.

С помощью функции HistoryDealGetTicket получаем тикет последней сделки и, используя свойство DEAL_COMMENT функции HistoryDealGetString, получаем комментарий к сделке. Если комментарий содержит sl, тогда это была сделка закрытия позиции по StopLoss.

flagStopLoss это глобальная переменная, которую мы теперь можем использовать в функции OnTick ():

```
bool flagStopLoss=false;

static datetime Old_TimeD1;
datetime New_TimeD1 [1];
bool IsNewBarD1=false;

int copiedD1=CopyTime (_Symbol, PERIOD_D1,0,1,New_TimeD1);
ResetLastError ();
if (copiedD1> 0)
{
    if (Old_TimeD1!=New_TimeD1 [0])
    {
        IsNewBarD1=true;
        Old_TimeD1=New_TimeD1 [0];
        flagStopLoss=false;
    }
}
else
{
    Print (GetLastError ());
    return;
}

if (IsNewBarD1==false)
{
    if (flagStopLoss==true) {
        return;
    }
}
```

Здесь эксперт прекращает вычисления, если получен StopLoss и не наступил новый дневной бар.

Так как для каждого финансового инструмента (символа) возможна только одна открытая позиция, в функции OnTick () нужно организовать проверку наличия открытой позиции, чтобы не пытаться открыть ее заново:

```
bool BuyOpened=false;
bool SellOpened=false;

if (PositionSelect (_Symbol) ==true)
{
    if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_BUY)
    {
        BuyOpened=true;
    }
    else if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_SELL)
    {
        SellOpened=true;
    }
}
```

```
}
```

Здесь функция PositionSelect копирует данные об открытой позиции символа в программное окружение. Затем, с помощью свойства POSITION_TYPE функции PositionGetInteger, выясняется, является ли открытая позиция позицией на продажу или покупку.

После выполнения различных проверок в функции OnTick () следуют вычисления сигналов торговой системы эксперта.

Как правило, для вычисления сигналов торговой системы требуются исторические данные символа. Сделать это можно с помощью функции CopyRates и структуры MqlRates, содержащей исторические цены бара символа:

```
MqlRates mrate [];  
ResetLastError ();  
if (CopyRates (Symbol (),Period (),0,3,mrate) <0)  
{  
    Print (GetLastError ());  
    return;  
}  
ArraySetAsSeries (mrate, true);
```

Здесь в массив структуры MqlRates копируются данные последних трех баров, а затем меняется порядок доступа к массиву.

Наконец, после вычислений сигналов торговой системы эксперта можно открывать или закрывать позиции.

Но перед совершением сделки было бы неплохо проверить корректность объема, с которым мы собираемся выйти на рынок:

```
if (Lot <SymbolInfoDouble (Symbol (),SYMBOL_VOLUME_MIN) ||Lot>  
SymbolInfoDouble (Symbol (),SYMBOL_VOLUME_MAX)) return;
```

Открытие и закрытие позиции, изменение объема открытой позиции, изменение значения Stop Loss и Take Profit у открытой позиции, установка, модификация и удаление отложенных ордеров, все это может быть сделано с помощью функции OrderSend:

```
bool OrderSend (  
    MqlTradeRequest& request, // структура запроса  
    MqlTradeResult& result // структура ответа  
);
```

Тип торговой операции, которую будет пытаться выполнить функция OrderSend, определяется структурой MqlTradeRequest:

```
struct MqlTradeRequest  
{  
    ENUM_TRADE_REQUEST_ACTIONS action;  
    // Тип выполняемого действия  
    ulong magic;  
    // Штамп эксперта (идентификатор magic number)  
    ulong order;  
    // Тикет ордера  
    string symbol;  
    // Имя торгового инструмента  
    double volume;  
    // Запрашиваемый объем сделки в лотах  
    double price;  
    // Цена
```

```
double stoplimit;  
// Уровень StopLimit ордера  
double sl;  
// Уровень Stop Loss ордера  
double tp;  
// Уровень Take Profit ордера  
ulong deviation;  
// Максимально приемлемое отклонение от запрашиваемой цены  
ENUM_ORDER_TYPE type;  
// Тип ордера  
ENUM_ORDER_TYPE_FILLING type_filling;  
// Тип ордера по исполнению  
ENUM_ORDER_TYPE_TIME type_time;  
// Тип ордера по времени действия  
datetime expiration;  
// Срок истечения ордера (для ордеров типа ORDER_TIME_SPECIFIED)  
string comment;  
// Комментарий к ордеру  
};
```

Первый параметр action структуры MqlTradeRequest определяет тип торговой операции функции OrderSend с помощью перечисления ENUM_TRADE_REQUEST_ACTIONS.

Это может быть немедленное совершение сделки на покупку или продажу (TRADE_ACTION_DEAL), изменение значений Stop Loss и Take Profit у открытой позиции (TRADE_ACTION_SLTP), установка отложенного ордера на покупку или продажу (TRADE_ACTION_PENDING), изменение параметров отложенного ордера (TRADE_ACTION_MODIFY), удаление отложенного ордера (TRADE_ACTION_REMOVE).

Если вы хотите совершить немедленную сделку на покупку или продажу, в этом случае тип исполнения ордера для данного финансового инструмента или символа определяется брокером.

Это может быть немедленное исполнение (Instant Execution), исполнение по запросу (Request Execution), исполнение по рынку (Market Execution), биржевое исполнение (Exchange Execution).

Выяснить тип исполнения ордера можно с помощью свойства SYMBOL_TRADE_EXEMODE функции SymbolInfoInteger:

```
Print (EnumToString ((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger  
(Symbol (),SYMBOL_TRADE_EXEMODE)));
```

Для немедленного исполнения (Instant Execution), исполнение рыночного ордера осуществляется по цене, которую вы предлагаете брокеру. Если брокер не может принять ордер по предложенным ценам, он предложит трейдеру новые цены исполнения, которые будут содержаться в структуре MqlTradeResult.

Для немедленного исполнения (Instant Execution), заполнение структуры MqlTradeRequest для ордера на покупку будет иметь следующий вид:

```
MqlTradeRequest mrequest;  
ZeroMemory (mrequest);  
MqlTick latest_price;  
if (!SymbolInfoTick (_Symbol, latest_price))  
{  
Alert («Ошибка получения последних котировок – ошибка:», GetLastError (),»!!»);
```

```
return;
}
if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest.volume = Lot;
    mrequest.price = NormalizeDouble(latest_price.ask,_Digits);
    mrequest.sl = NormalizeDouble(latest_price.bid - 0.01,_Digits);
    mrequest.tp = NormalizeDouble(latest_price.ask +0.01,_Digits);
    mrequest.deviation=10;
    mrequest.type = ORDER_TYPE_BUY;
    mrequest.type_filling = ORDER_FILLING_FOK;
}
```

Здесь с помощью функции SymbolInfoTick в структуру MqlTick получают текущие цены символа для предложения их брокеру.

Для немедленного исполнения (Instant Execution) заполняются обязательные поля структуры MqlTradeRequest action, symbol, volume, price, sl, tp, deviation, type, type_filling.

На практике, максимально приемлемое отклонение от запрашиваемой цены deviation, задаваемое в пунктах, которое принимается брокером, не более 5 пунктов. При сильном движении рынка, при поступлении ордера брокеру, если цена ушла на большее значение, произойдет так называемое «Перекокотирование» (Requote) – брокер вернет цены, по которым может быть исполнен данный ордер.

Функция NormalizeDouble здесь используется для округления цен до количества десятичных знаков после запятой, определяющего точность измерения цены символа текущего графика.

Для ордера на продажу заполнение Instant Execution обязательных полей структуры MqlTradeRequest будет иметь следующий вид:

```
if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest.volume = Lot;
    mrequest.price = NormalizeDouble(latest_price.ask,_Digits);
    mrequest.sl = NormalizeDouble(latest_price.ask +0.01,_Digits);
    mrequest.tp = NormalizeDouble(latest_price.bid - 0.01,_Digits);
    mrequest.deviation=10;
    mrequest.type = ORDER_TYPE_SELL;
    mrequest.type_filling = ORDER_FILLING_FOK;
}
```

После заполнения полей структуры MqlTradeRequest рекомендуется проверить ее с помощью функции OrderCheck:

```
bool OrderCheck (
    MqlTradeRequest& request, // структура запроса
    MqlTradeCheckResult& result // структура ответа
);
```

Результаты проверки будут содержаться в структуре MqlTradeCheckResult:

```
struct MqlTradeCheckResult
{
```

```
uint retcode;  
// Код ответа  
double balance;  
// Баланс после совершения сделки  
double equity;  
// Значение собственных средств после совершения сделки  
double profit;  
// Плавающая прибыль  
double margin;  
// Маржевые требования  
double margin_free;  
// Свободная маржа  
double margin_level;  
// Уровень маржи  
string comment;  
// Комментарий к коду ответа (описание ошибки)  
};
```

Функция OrderCheck возвращает true в случае успешной проверки структуры MqlTradeRequest, при этом код retcode будет равен 0, в противном случае функция вернет false:

```
MqlTradeCheckResult check_result;  
ZeroMemory (check_result);  
if (!OrderCheck (mrequest, check_result))  
{  
    if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);  
    if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);  
    if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);  
    if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-  
нения запроса»);  
    return;  
}
```

После проверки структуры MqlTradeRequest можно отсылать запрос на совершение торговой операции брокеру, используя функцию OrderSend:

```
// —————  
MqlTradeRequest mrequest;  
MqlTradeCheckResult check_result;  
MqlTradeResult mresult;  
  
MqlTick latest_price;  
if (!SymbolInfoTick (_Symbol, latest_price))  
{  
    Alert («Ошибка получения последних котировок – ошибка:», GetLastError (),»!!»);  
    return;  
}  
if (TradeSignalBuy==true&&BuyOpened==false) {  
    if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol  
(_),SYMBOL_TRADE_EXEMODE))==SYMBOL_TRADE_EXECUTION_INSTANT) {  
        ZeroMemory (mrequest);  
        mrequest.action = TRADE_ACTION_DEAL;
```

```
mrequest.symbol = _Symbol;
mrequest.volume = Lot;
mrequest.price = NormalizeDouble(latest_price.ask, _Digits);
mrequest.sl = NormalizeDouble(latest_price.bid - 0.01, _Digits);
mrequest.tp = NormalizeDouble(latest_price.ask + 0.01, _Digits);
mrequest.deviation=10;
mrequest.type = ORDER_TYPE_BUY;
mrequest.type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
    if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
    if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
    if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
    if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
    return;
} else {
    if (OrderSend (mrequest, mresult)) {
        if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
        {
            Print («Price», mresult.price);
        }
        else
        {
            if(mresult.retcode==10004) //Пеквота
            {
                Print («Requote bid ",mresult.bid);
                Print («Requote ask ",mresult.ask);
            } else {
                Print («Retcode ",mresult.retcode);
            }
        }
    } else {
        Print («Retcode ",mresult.retcode);
    }
}
}
}
// -----
if (TradeSignalSell==true&&SellOpened==false) {
    if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
        ZeroMemory (mrequest);
        mrequest.action = TRADE_ACTION_DEAL;
        mrequest.symbol = _Symbol;
```

```
mrequest.volume = Lot;
mrequest.price = NormalizeDouble(latest_price.ask, _Digits);
mrequest.sl = NormalizeDouble(latest_price.ask + 0.01, _Digits);
mrequest.tp = NormalizeDouble(latest_price.bid - 0.01, _Digits);
mrequest.deviation=10;
mrequest.type = ORDER_TYPE_SELL;
mrequest.type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
    if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
    if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
    if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
    if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
    return;
} else {
    if (OrderSend (mrequest, mresult)) {
        if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
        {
            Print («Price», mresult.price);
        }
        else
        {
            if(mresult.retcode==10004) //Пеквота
            {
                Print («Requote bid ",mresult.bid);
                Print («Requote ask ",mresult.ask);
            } else {
                Print («Retcode ",mresult.retcode);
            }
        }
        } else {
            Print («Retcode ",mresult.retcode);
        }
    }
}
// -----
```

Здесь запрос на совершение торговой операции отсылается, если есть сигнал на открытие позиции и при этом открытая позиция еще не существует.

После проверки OrderCheck производится повторная проверка структуры MqlTradeRequest в виде возвращаемого значения функции OrderSend. Далее выполняется проверка кода результата операции структуры MqlTradeResult.

Исполнение ордера по запросу (Request Execution) я лично не встречал у брокеров. Вместо немедленного исполнения (Instant Execution) брокер может предложить исполнение ордера по рынку (Market Execution) или биржевое исполнение (Exchange Execution).

В режиме исполнения по рынку (Market Execution) сделка совершается по цене, предложенной брокером, при этом реквоты отсутствуют. В режиме биржевого исполнения (Exchange Execution) торговые операции якобы выводятся во внешнюю торговую систему и сделки выполняются по текущим рыночным ценам, при этом реквоты также отсутствуют.

При исполнении по рынку (Market Execution) или биржевом исполнении (Exchange Execution) обязательными для заполнения являются поля структуры MqlTradeRequest action, symbol, volume, type, type_filling:

```
// -----
if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE))==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
//или SYMBOL_TRADE_EXECUTION_MARKET
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_DEAL;
mrequest.symbol = _Symbol;
mrequest.volume = Lot;
mrequest.type = ORDER_TYPE_BUY;
mrequest.type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
return;
} else {
if (OrderSend (mrequest, mresult)) {
if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
{
// -----
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_SLTP;
mrequest.symbol = _Symbol;
mrequest.sl = NormalizeDouble(mresult.price - 0.01,_Digits);
mrequest.tp = NormalizeDouble(mresult.price +0.01,_Digits);
ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
return;
} else {
if (OrderSend (mrequest, mresult)) {
```

```
        if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
        {
            Print («SL», mrequest.sl, «TP ",mrequest.tp);
        }
        else
        {
            Print («Retcode ",mresult.retcod);
        }
        } else {
            Print («Retcode ",mresult.retcod);
        }
        }
        // -----
        }
        else
        {
            Print («Retcode ",mresult.retcod);
        }
        } else {
            Print («Retcode ",mresult.retcod);
        }
        }
        }
        // -----
        if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
            //или SYMBOL_TRADE_EXECUTION_MARKET
            ZeroMemory (mrequest);
            mrequest.action = TRADE_ACTION_DEAL;
            mrequest.symbol = _Symbol;
            mrequest. volume = Lot;
            mrequest. type = ORDER_TYPE_SELL;
            mrequest. type_filling = ORDER_FILLING_FOK;

            ZeroMemory (check_result);
            ZeroMemory (mresult);
            if (!OrderCheck (mrequest, check_result))
            {
                if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
                if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
                return;
            } else {
                if (OrderSend (mrequest, mresult)) {
                    if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
                    {
                        // -----
```

```
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_SLTP;
mrequest.symbol = _Symbol;
mrequest.tp = NormalizeDouble(mresult.price - 0.01, _Digits);
mrequest.sl = NormalizeDouble(mresult.price + 0.01, _Digits);
ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcod==10015) Alert («Неправильная цена в запросе»);
if(check_result.retcod==10016) Alert («Неправильные стопы в запросе»);
return;
} else {
if (OrderSend (mrequest, mresult)) {
if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
{
Print («SL», mrequest.sl, «TP ",mrequest.tp);
}
else
{
Print («Retcode ",mresult.retcod);
}
} else {
Print («Retcode ",mresult.retcod);
}
}
// -----
}
else
{
Print («Retcode ",mresult.retcod);
}
} else {
Print («Retcode ",mresult.retcod);
}
}
}
// -----
```

Здесь позиция открывается без определения StopLoss и TakeProfit, а затем, в случае успешного выполнения запроса, размещается торговый приказ на модификацию уровней StopLoss и TakeProfit.

Для установки отложенного ордера на покупку или продажу (TRADE_ACTION_PENDING), требуется указание 11 полей структуры MqlTradeRequest: action, symbol, volume, price, stoplimit, sl, tp, type, type_filling, type_time, expiration. При этом поле type может принимать значения:

ORDER_TYPE_BUY_LIMIT – отложенный ордер на покупку, при этом текущие цены выше цены ордера.

ORDER_TYPE_SELL_LIMIT – отложенный ордер на продажу, при этом текущие цены ниже цены ордера.

ORDER_TYPE_BUY_STOP – отложенный ордер на покупку, при этом текущие цены ниже цены ордера.

ORDER_TYPE_SELL_STOP – отложенный ордер на продажу, при этом текущие цены выше цены ордера.

ORDER_TYPE_BUY_STOP_LIMIT – отложенное выставление отложенного ордера типа Buy Limit для торговли на откате, при этом текущие цены ниже цены ордера.

ORDER_TYPE_SELL_STOP_LIMIT – отложенное выставление отложенного ордера типа Sell Limit для торговли на откате, при этом текущие цены выше цены ордера.

Для изменения параметров отложенного ордера (TRADE_ACTION_MODIFY), требуется указание 7 полей структуры MqlTradeRequest: action, order, price, sl, tp, type_time, expiration. При этом значение поля order берется из структуры MqlTradeResult результата выставления ордера.

Для удаления отложенного ордера (TRADE_ACTION_REMOVE), требуется указание 2 полей структуры MqlTradeRequest: action и order.

Пример создания эксперта

В качестве основы советника возьмем следующий код:

```
input double Lot=1;
input int EA_Magic=1000;
input double spreadLevel=5.0;
input double StopLoss=0.01;
input double Profit=0.01;

bool flagStopLoss=false;

int OnCheckTradeInit () {
    //Проверка запуска эксперта на реальном счете
    if ((ENUM_ACCOUNT_TRADE_MODE) AccountInfoInteger
        (ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL) {
        int mb=MessageBox («Запустить советник на реальном счете?», «Message Box»,
            MB_YESNO|MB_ICONQUESTION);
        if (mb==IDNO) return (0);
    }
    // -----
    //Проверки: запрещена торговля в случае подключения к счету в режиме инвестора,
    //отсутствия соединения к серверу, запрета торговли на стороне сервера, если счет
отправлен в архив
    //брокер запрещает автоматическую торговлю

    if (!TerminalInfoInteger (TERMINAL_CONNECTED)) {
        Alert («No connection to the trade server»);
        return (0);
    } else {
        if (!AccountInfoInteger (ACCOUNT_TRADE_ALLOWED)) {
            Alert («Trade for this account is prohibited»);
            return (0);
        }
    }
    if (!AccountInfoInteger (ACCOUNT_TRADE_EXPERT)) {
        Alert («Trade with the help of experts for the account is prohibited»);
        return (0);
    }

    // -----

    //Проверить корректность объема, с которым мы собираемся выйти на рынок
    if (Lot <SymbolInfoDouble (Symbol (),SYMBOL_VOLUME_MIN) ||Lot>
SymbolInfoDouble (Symbol (),SYMBOL_VOLUME_MAX)) {
        Alert («Lot is not correct!!!»);
        return (0);
    }
    return (INIT_SUCCEEDED);
}
```

```
    }

    //+-----+
    //| Expert initialization function |
    //+-----+
    int OnInit ()
    {

        return (OnCheckTradeInit ());
    }
    //+-----+
    //| Expert deinitialization function |
    //+-----+
    void OnDeinit (const int reason)
    {

    }

    int OnCheckTradeTick () {

        //Проверка отсутствия соединения к серверу
        if (!TerminalInfoInteger (TERMINAL_CONNECTED)) {
            Alert («No connection to the trade server»);
            return (0);
        }

        //Включена ли кнопка авто-торговли в клиентском терминале
        if (!TerminalInfoInteger (TERMINAL_TRADE_ALLOWED)) {
            Alert («Разрешение на автоматическую торговлю выключено!»);
            return (0);
        }

        //Разрешение на торговлю с помощью эксперта отключено в общих свойствах самого
эксперта
        if (!MQLInfoInteger (MQL_TRADE_ALLOWED)) {
            Alert («Автоматическая торговля запрещена в свойствах эксперта», __FILE__);
            return (0);
        }

        //-----

        //Уровень залоговых средств, при котором требуется пополнение счета
        if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger
(AACCOUNT_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_PERCENT) {
            if (AccountInfoDouble (ACCOUNT_MARGIN_LEVEL) != 0 && AccountInfoDouble
(AACCOUNT_MARGIN_LEVEL) <= AccountInfoDouble
(AACCOUNT_MARGIN_SO_CALL)) {
```

```
Alert («Margin Call!!!»);  
return (0);  
}}
```

```
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger  
(ACCOUNT_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_MONEY) {  
    if (AccountInfoDouble (ACCOUNT_EQUITY) <= AccountInfoDouble  
(ACCOUNT_MARGIN_SO_CALL)) {  
        Alert («Margin Call!!!»);  
        return (0);  
    }  
}
```

//Уровень залоговых средств, при достижении которого происходит принудительное
закрытие самой убыточной позиции

```
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger  
(ACCOUNT_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_PERCENT) {  
    if (AccountInfoDouble (ACCOUNT_MARGIN_LEVEL) != 0 && AccountInfoDouble  
(ACCOUNT_MARGIN_LEVEL) <= AccountInfoDouble (ACCOUNT_MARGIN_SO_SO)) {  
        Alert («Stop Out!!!»);  
        return (0);  
    }  
}
```

```
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger  
(ACCOUNT_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_MONEY) {  
    if (AccountInfoDouble (ACCOUNT_EQUITY) <= AccountInfoDouble  
(ACCOUNT_MARGIN_SO_SO)) {  
        Alert («Stop Out!!!»);  
        return (0);  
    }  
}
```

```
// -----  
-----
```

//Проверка размера свободных средств на счете, доступных для открытия позиции

```
double margin;  
MqlTick last_tick;  
ResetLastError ();  
if (SymbolInfoTick (Symbol (), last_tick))  
{  
    if (OrderCalcMargin (ORDER_TYPE_BUY, Symbol (), Lot, last_tick.ask, margin))  
    {  
        if (margin > AccountInfoDouble (ACCOUNT_MARGIN_FREE)) {  
            Alert («Not enough money in the account!»);  
            return (0);  
        }  
    }  
}  
else  
{
```

```
Print (GetLastError ());
}

// -----

//Контроль над спредом брокера
double _spread=SymbolInfoInteger (Symbol (),SYMBOL_SPREAD) *MathPow (10, -
SymbolInfoInteger (Symbol (),SYMBOL_DIGITS)) /MathPow (10, -4);

if (_spread> spreadLevel) {
Alert («Слишком большой спред!»);
return (0);
}

// -----

//Проверка ограничений на торговые операции по символу, установленные брокером
if ((ENUM_SYMBOL_TRADE_MODE) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_MODE)!=SYMBOL_TRADE_MODE_FULL) {
Alert («Установлены ограничения на торговые операции»);
return (0);
}

// -----

//Достаточно ли баров в истории для расчета советника
if (Bars (Symbol (), 0) <100)
{
Alert («In the chart little bars, Expert will not work!!»);
return (0);
}

return (1);
}

//+-----+
//| Expert tick function |
//+-----+
void OnTick ()
{

if (!OnCheckTradeTick ()) {
return;
}

// -----
-----

//Ограничить вычисления советника по появлению нового бара на графике
```

```
static datetime last_time;
datetime last_bar_time= (datetime) SeriesInfoInteger (Symbol (),Period
(),SERIES_LASTBAR_DATE);
if (last_time!=last_bar_time)
{
last_time=last_bar_time;
} else {
return;
}

//Ограничить вычисления советника по flagStopLoss
static datetime last_time_daily;
datetime last_bar_time_daily= (datetime) SeriesInfoInteger (Symbol
(),PERIOD_D1,SERIES_LASTBAR_DATE);
if (last_time_daily!=last_bar_time_daily)
{
last_time_daily=last_bar_time_daily;
flagStopLoss=false;
}

if (flagStopLoss==true) return;
// -----

//Проверка наличия открытой позиции, чтобы не пытаться открыть ее заново
bool BuyOpened=false;
bool SellOpened=false;

if (PositionSelect (_Symbol) ==true)
{
if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_BUY)
{
BuyOpened=true;
}
else if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_SELL)
{
SellOpened=true;
}
}

// -----

//Для вычисления сигналов торговой системы требуются исторические данные сим-
вола

MqlRates mrate [];
ResetLastError ();
if (CopyRates (Symbol (),Period (),0,3,mrate) <0)
{
Print (GetLastError ());
}
```

```
return;
}

ArraySetAsSeries (mrate, true);

// -----

bool TradeSignalBuy=false;
bool TradeSignalSell=false;

// -----

MqlTradeRequest mrequest;
MqlTradeCheckResult check_result;
MqlTradeResult mresult;

MqlTick latest_price;
if (!SymbolInfoTick (_Symbol, latest_price))
{
Alert («Ошибка получения последних котировок – ошибка:», GetLastError (),»!!»);
return;
}
if (TradeSignalBuy==true&&BuyOpened==false) {
// -----

if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE))==SYMBOL_TRADE_EXECUTION_INSTANT) {
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_DEAL;
mrequest.symbol = _Symbol;
mrequest.volume = Lot;
mrequest.price = NormalizeDouble(latest_price.ask,_Digits);
mrequest.sl = NormalizeDouble(latest_price.bid – StopLoss,_Digits);
mrequest.tp = NormalizeDouble(latest_price.ask + Profit,_Digits);
mrequest.deviation=10;
mrequest.type = ORDER_TYPE_BUY;
mrequest.type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
return;
}
```

```
    } else {
    if (OrderSend (mrequest, mresult)) {
        if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
        {
            Print («Price», mresult.price);
        }
        else
        {
            if(mresult.retcod==10004) //Реквота
            {
                Print («Requote bid ",mresult.bid);
                Print («Requote ask ",mresult.ask);
            } else {
                Print («Retcode ",mresult.retcod);
            }
        }
    } else {
        Print («Retcode ",mresult.retcod);
    }
}
// -----

if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE))==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest. volume = Lot;
    mrequest. type = ORDER_TYPE_BUY;
    mrequest. type_filling = ORDER_FILLING_FOK;

    ZeroMemory (check_result);
    ZeroMemory (mresult);
    if (!OrderCheck (mrequest, check_result))
    {
        if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
        if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
        return;
    } else {
        if (OrderSend (mrequest, mresult)) {
            if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
            {
                // -----
                ZeroMemory (mrequest);
                mrequest.action = TRADE_ACTION_SLTP;
```



```
mrequest.volume = Lot;
mrequest.price = NormalizeDouble(latest_price.bid,_Digits);
mrequest.sl = NormalizeDouble(latest_price.ask + StopLoss,_Digits);
mrequest.tp = NormalizeDouble(latest_price.bid - Profit,_Digits);
mrequest.deviation=10;
mrequest.type = ORDER_TYPE_SELL;
mrequest.type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
    if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
    if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
    if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
    if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
    return;
} else {
    if (OrderSend (mrequest, mresult)) {
        if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
        {
            Print («Price», mresult.price);
        }
        else
        {
            if(mresult.retcode==10004) //Пеквота
            {
                Print («Requote bid ",mresult.bid);
                Print («Requote ask ",mresult.ask);
            } else {
                Print («Retcode ",mresult.retcode);
            }
        }
        } else {
            Print («Retcode ",mresult.retcode);
        }
    }
}

// -----
-----

if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest.volume = Lot;
    mrequest.type = ORDER_TYPE_SELL;
```

```
mrequest.type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
    if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
    if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
    return;
} else {
    if (OrderSend (mrequest, mresult)) {
        if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
        {
            // —————
            ZeroMemory (mrequest);
            mrequest.action = TRADE_ACTION_SLTP;
            mrequest.symbol = _Symbol;
            mrequest.tp = NormalizeDouble(mresult.price – Profit,_Digits);
            mrequest.sl = NormalizeDouble(mresult.price + StopLoss,_Digits);
            ZeroMemory (check_result);
            ZeroMemory (mresult);
            if (!OrderCheck (mrequest, check_result))
            {
                if(check_result.retcod==10015) Alert («Неправильная цена в запросе»);
                if(check_result.retcod==10016) Alert («Неправильные стопы в запросе»);
                return;
            } else {
                if (OrderSend (mrequest, mresult)) {
                    if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
                    {
                        Print («SL», mrequest.sl, «TP ",mrequest.tp);
                    }
                    else
                    {
                        Print («Retcode ",mresult.retcod);
                    }
                } else {
                    Print («Retcode ",mresult.retcod);
                }
            }
            // —————
        }
        else
        {
            Print («Retcode ",mresult.retcod);
        }
    }
}
```

```
    } else {  
    Print («Retcode ",mresult.retcode);  
    }  
    }  
    }  
    // -----  
-----
```

```
}
```

```
// -----  
}
```

```
//+-----+  
//| Trade function |  
//+-----+  
void OnTrade ()  
{  
static int _deals;  
ulong _ticket=0;
```

```
if (HistorySelect (0,TimeCurrent ()))  
{  
int i=HistoryDealsTotal () -1;
```

```
if (_deals!=i) {  
_deals=i;  
} else {return;}
```

```
if ((_ticket=HistoryDealGetTicket (i))> 0)  
{  
string _comment=HistoryDealGetString (_ticket, DEAL_COMMENT);  
if (StringFind (_comment,«sl», 0)!=-1) {  
flagStopLoss=true;  
}  
}  
}  
}
```

Здесь общие проверки функции OnInit выделены в отдельную функцию OnCheckTradeInit, а общие проверки функции OnTick выделены в отдельную функцию OnCheckTradeTick.

Советник будет отправлять ордера на покупку и продажу при установке флагов TradeSignalBuy и TradeSignalSell в значение true.

Установку значений флагов TradeSignalBuy и TradeSignalSell должна осуществлять торговая система.

В качестве торговой системы возьмем «Метод Сидуса»:

Валютные пары: EUR/GBP и EUR/USD.

Временной интервал – H1;

Экспоненциальные скользящие средние (Exponential Moving Average): 18 ЕМА и 28 ЕМА;

Weighted Moving Average – 5WMA и 8 WMA.

Торговые сигналы на вход в рынок по Методу Сидуса:

Открытие позиции на покупку: 5WMA и 8 WMA скользящие средние пересекают туннель из 18 ЕМА и 28 ЕМА снизу вверх.

Открытие позиции на продажу: 5WMA и 8 WMA скользящие средние пересекают туннель из 18 ЕМА и 28 ЕМА сверху вниз.

Торговые сигналы на выход из рынка по Методу Сидуса:

На покупку: цена на графике достигла вершины и 5 WMA как бы «ныряет» под 8 WMA скользящую среднюю. Следует закрыть открытую торговую позицию.

На продажу: цена на графике достигла дна и скользящая средняя 5 WMA как бы «прыгает» над 8WMA. Следует закрыть торговую позицию.

Код советника, реализующий Метод Сидуса:

```
input double Lot=1;
input int EA_Magic=1000;
input double spreadLevel=5.0;
input double StopLoss=0.01;
input double Profit=0.01;
input int numberBarOpenPosition=5;
input int numberBarStopPosition=5;
```

```
bool flagStopLoss=false;
```

```
int handleIMA18;
double MA18Buffer [];
int handleIMA28;
double MA28Buffer [];
int handleIWMA5;
double WMA5Buffer [];
int handleIWMA8;
double WMA8Buffer [];
```

```
int OnCheckTradeInit () {
//Проверка запуска эксперта на реальном счете
if ((ENUM_ACCOUNT_TRADE_MODE) AccountInfoInteger
(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL) {
    int mb=MessageBox («Запустить советник на реальном счете?», «Message Box»,
    MB_YESNO|MB_ICONQUESTION);
    if (mb==IDNO) return (0);
}
// -----
//Проверки: запрещена торговля в случае подключения к счету в режиме инвестора,
//отсутствия соединения к серверу, запрета торговли на стороне сервера, если счет
отправлен в архив
//брокер запрещает автоматическую торговлю

if (!TerminalInfoInteger (TERMINAL_CONNECTED)) {
    Alert («No connection to the trade server»);
```

```
return (0);
} else {
if (!AccountInfoInteger (ACCOUNT_TRADE_ALLOWED)) {
Alert («Trade for this account is prohibited»);
return (0);
}
}
if (!AccountInfoInteger (ACCOUNT_TRADE_EXPERT)) {
Alert («Trade with the help of experts for the account is prohibited»);
return (0);
}

//-----

//Проверить корректность объема, с которым мы собираемся выйти на рынок
if (Lot <SymbolInfoDouble (Symbol (),SYMBOL_VOLUME_MIN) ||Lot>
SymbolInfoDouble (Symbol (),SYMBOL_VOLUME_MAX)) {
Alert («Lot is not correct!!!»);
return (0);
}
return (INIT_SUCCEEDED);

}

//+-----+
//| Expert initialization function |
//+-----+
int OnInit ()
{

handleIMA18=iMA (_Symbol, PERIOD_H1,18,0,MODE_EMA, PRICE_CLOSE);
handleIMA28=iMA (_Symbol, PERIOD_H1,28,0,MODE_EMA, PRICE_CLOSE);
handleIWMA5=iMA (_Symbol, PERIOD_H1,5,0,MODE_LWMA, PRICE_CLOSE);
handleIWMA8=iMA (_Symbol, PERIOD_H1,8,0,MODE_LWMA, PRICE_CLOSE);

return (OnCheckTradeInit ());
}
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit (const int reason)
{

}

int OnCheckTradeTick () {

//Проверка отсутствия соединения к серверу
if (!TerminalInfoInteger (TERMINAL_CONNECTED)) {
```

```
Alert («No connection to the trade server»);
return (0);
}

//Включена ли кнопка авто-торговли в клиентском терминале
if (!TerminalInfoInteger (TERMINAL_TRADE_ALLOWED)) {
Alert («Разрешение на автоматическую торговлю выключено!»);
return (0);
}

//Разрешение на торговлю с помощью эксперта отключено в общих свойствах самого
эксперта
if (!MQLInfoInteger (MQL_TRADE_ALLOWED)) {
Alert («Автоматическая торговля запрещена в свойствах эксперта», __FILE__);
return (0);
}

// -----

//Уровень залоговых средств, при котором требуется пополнение счета
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger
(AACCOUNT_MARGIN_SO_MODE) ==ACCOUNT_STOPOUT_MODE_PERCENT) {
if (AccountInfoDouble (ACCOUNT_MARGIN_LEVEL) != 0 && AccountInfoDouble
(AACCOUNT_MARGIN_LEVEL) <= AccountInfoDouble
(AACCOUNT_MARGIN_SO_CALL)) {
Alert («Margin Call!!!»);
return (0);
}}

if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger
(AACCOUNT_MARGIN_SO_MODE) ==ACCOUNT_STOPOUT_MODE_MONEY) {
if (AccountInfoDouble (ACCOUNT_EQUITY) <= AccountInfoDouble
(AACCOUNT_MARGIN_SO_CALL)) {
Alert («Margin Call!!!»);
return (0);
}}

//Уровень залоговых средств, при достижении которого происходит принудительное
закрытие самой убыточной позиции
if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger
(AACCOUNT_MARGIN_SO_MODE) ==ACCOUNT_STOPOUT_MODE_PERCENT) {
if (AccountInfoDouble (ACCOUNT_MARGIN_LEVEL) != 0 && AccountInfoDouble
(AACCOUNT_MARGIN_LEVEL) <= AccountInfoDouble (ACCOUNT_MARGIN_SO_SO)) {
Alert («Stop Out!!!»);
return (0);
}}
```

```
        if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger
(Account_MARGIN_SO_MODE) == ACCOUNT_STOPOUT_MODE_MONEY) {
        if (AccountInfoDouble (ACCOUNT_EQUITY) <= AccountInfoDouble
(Account_MARGIN_SO_SO)) {
            Alert («Stop Out!!!»);
            return (0);
        }
    }

    // -----

    //Проверка размера свободных средств на счете, доступных для открытия позиции
    double margin;
    MqlTick last_tick;
    ResetLastError ();
    if (SymbolInfoTick (Symbol (),last_tick))
    {
        if (OrderCalcMargin (ORDER_TYPE_BUY,Symbol(),Lot,last_tick.ask, margin))
        {
            if (margin> AccountInfoDouble (ACCOUNT_MARGIN_FREE)) {
                Alert («Not enough money in the account!»);
                return (0);
            }
        }
    }
    else
    {
        Print (GetLastError ());
    }

    // -----

    //Контроль над спредом брокера
    double _spread=SymbolInfoInteger (Symbol (),SYMBOL_SPREAD) *MathPow (10, -
SymbolInfoInteger (Symbol (),SYMBOL_DIGITS)) /MathPow (10, -4);

    if (_spread> spreadLevel) {
        Alert («Слишком большой спред!»);
        return (0);
    }

    // -----

    //Проверка ограничений на торговые операции по символу, установленные брокером
    if ((ENUM_SYMBOL_TRADE_MODE) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_MODE)!=SYMBOL_TRADE_MODE_FULL) {
        Alert («Установлены ограничения на торговые операции»);
        return (0);
    }
}
```

```
// -----  
  
//Достаточно ли баров в истории для расчета советника  
if (Bars (Symbol (), 0) <100)  
{  
Alert («In the chart little bars, Expert will not work!!»);  
return (0);  
}  
  
return (1);  
}  
  
//+-----+  
//| Expert tick function |  
//+-----+  
void OnTick ()  
{  
  
if (!OnCheckTradeTick ()) {  
return;  
}  
  
// -----  
-----  
  
//Ограничить вычисления советника по появлению нового бара на графике  
static datetime last_time;  
datetime last_bar_time= (datetime) SeriesInfoInteger (Symbol (),Period  
(0,SERIES_LASTBAR_DATE));  
if (last_time!=last_bar_time)  
{  
last_time=last_bar_time;  
} else {  
return;  
}  
  
//Ограничить вычисления советника по flagStopLoss  
static datetime last_time_daily;  
datetime last_bar_time_daily= (datetime) SeriesInfoInteger (Symbol  
(0,PERIOD_D1,SERIES_LASTBAR_DATE);  
if (last_time_daily!=last_bar_time_daily)  
{  
last_time_daily=last_bar_time_daily;  
flagStopLoss=false;  
}  
  
if (flagStopLoss==true) return;  
// -----
```

```
//Проверка наличия открытой позиции, чтобы не пытаться открыть ее заново
bool BuyOpened=false;
bool SellOpened=false;

if (PositionSelect (_Symbol) ==true)
{
if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_BUY)
{
BuyOpened=true;
}
else if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_SELL)
{
SellOpened=true;
}
}

//-----

//Для вычисления сигналов торговой системы требуются исторические данные сим-
вола

MqlRates mrate [];
ResetLastError ();
if (CopyRates (Symbol (),Period (),0,numberBarStopPosition, mrate) <0)
{
Print (GetLastError ());
return;
}

ArraySetAsSeries (mrate, true);

//-----

bool TradeSignalBuy=false;
bool TradeSignalSell=false;

TradeSignalBuy=OnTradeSignalBuy ();
TradeSignalSell=OnTradeSignalSell ();

bool TradeSignalBuyStop=false;
bool TradeSignalSellStop=false;

TradeSignalBuyStop=OnTradeSignalBuyStop (mrate);
TradeSignalSellStop=OnTradeSignalSellStop (mrate);

//-----
```

```
MqlTradeRequest mrequest;
MqlTradeCheckResult check_result;
MqlTradeResult mresult;

MqlTick latest_price;
if (!SymbolInfoTick (_Symbol, latest_price))
{
    Alert («Ошибка получения последних котировок – ошибка:», GetLastError (),»!!»);
    return;
}
if (TradeSignalBuy==true&&BuyOpened==false) {
// -----
if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest.volume = Lot;
    mrequest.price = NormalizeDouble(latest_price.ask,_Digits);
    mrequest.sl = NormalizeDouble(latest_price.bid – StopLoss,_Digits);
    mrequest.tp = NormalizeDouble(latest_price.ask + Profit,_Digits);
    mrequest.deviation=10;
    mrequest.type = ORDER_TYPE_BUY;
    mrequest.type_filling = ORDER_FILLING_FOK;

    ZeroMemory (check_result);
    ZeroMemory (mresult);
    if (!OrderCheck (mrequest, check_result))
    {
        if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
        if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
        if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
        if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
        return;
    } else {
        if (OrderSend (mrequest, mresult)) {
            if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
            {
                Print («Price», mresult.price);
            }
            else
            {
                if(mresult.retcode==10004) //Пеквота
                {
                    Print («Requote bid ",mresult.bid);
                    Print («Requote ask ",mresult.ask);
```

```
    } else {
    Print («Retcode ",mresult.retcod);
    }
    }
    } else {
    Print («Retcode ",mresult.retcod);
    }
    }
    }
    // -----
-----
    if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE))==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest. volume = Lot;
    mrequest. type = ORDER_TYPE_BUY;
    mrequest. type_filling = ORDER_FILLING_FOK;

    ZeroMemory (check_result);
    ZeroMemory (mresult);
    if (!OrderCheck (mrequest, check_result))
    {
    if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
    if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
    return;
    } else {
    if (OrderSend (mrequest, mresult)) {
    if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
    {
    // -----
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_SLTP;
    mrequest.symbol = _Symbol;
    mrequest.sl = NormalizeDouble(mresult.price – StopLoss,_Digits);
    mrequest.tp = NormalizeDouble(mresult.price + Profit,_Digits);
    ZeroMemory (check_result);
    ZeroMemory (mresult);
    if (!OrderCheck (mrequest, check_result))
    {
    if(check_result.retcod==10015) Alert («Неправильная цена в запросе»);
    if(check_result.retcod==10016) Alert («Неправильные стопы в запросе»);
    return;
    } else {
    if (OrderSend (mrequest, mresult)) {
```



```
        if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
        if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
        if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
        if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
        return;
    } else {
        if (OrderSend (mrequest, mresult)) {
            if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
            {
                Print («Price», mresult.price);
            }
            else
            {
                if(mresult.retcode==10004) //Пеквота
                {
                    Print («Requote bid ",mresult.bid);
                    Print («Requote ask ",mresult.ask);
                } else {
                    Print («Retcode ",mresult.retcode);
                }
            }
        } else {
            Print («Retcode ",mresult.retcode);
        }
    }
}
// -----

if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest.volume = Lot;
    mrequest.type = ORDER_TYPE_BUY;
    mrequest.type_filling = ORDER_FILLING_FOK;

    ZeroMemory (check_result);
    ZeroMemory (mresult);
    if (!OrderCheck (mrequest, check_result))
    {
        if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
        if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
        return;
    } else {
        if (OrderSend (mrequest, mresult)) {
```

```
        if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
        {
            Print («Price», mresult.price);
        }
        else
        {
            Print («Retcode ",mresult.retcod);
        }
        } else {
            Print («Retcode ",mresult.retcod);
        }
        }
        }
        // -----
    }
    // -----

    if (TradeSignalSell==true&&SellOpened==false) {
        // -----

        if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
            ZeroMemory (mrequest);
            mrequest.action = TRADE_ACTION_DEAL;
            mrequest.symbol = _Symbol;
            mrequest.volume = Lot;
            mrequest.price = NormalizeDouble(latest_price.bid,_Digits);
            mrequest.sl = NormalizeDouble(latest_price.ask + StopLoss,_Digits);
            mrequest.tp = NormalizeDouble(latest_price.bid – Profit,_Digits);
            mrequest.deviation=10;
            mrequest.type = ORDER_TYPE_SELL;
            mrequest.type_filling = ORDER_FILLING_FOK;

            ZeroMemory (check_result);
            ZeroMemory (mresult);
            if (!OrderCheck (mrequest, check_result))
            {
                if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
                if(check_result.retcod==10015) Alert («Неправильная цена в запросе»);
                if(check_result.retcod==10016) Alert («Неправильные стопы в запросе»);
                if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
                return;
            } else {
                if (OrderSend (mrequest, mresult)) {
                    if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
```

```
{
Print («Price», mresult.price);
}
else
{
if(mresult.retcod==10004) //Реквот
{
Print («Requote bid ",mresult.bid);
Print («Requote ask ",mresult.ask);
} else {
Print («Retcode ",mresult.retcod);
}
}
} else {
Print («Retcode ",mresult.retcod);
}
}
}
// -----
-----
if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_DEAL;
mrequest.symbol = _Symbol;
mrequest. volume = Lot;
mrequest. type = ORDER_TYPE_SELL;
mrequest. type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
return;
} else {
if (OrderSend (mrequest, mresult)) {
if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
{
// -----
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_SLTP;
mrequest.symbol = _Symbol;
mrequest.tp = NormalizeDouble(mresult.price – Profit,_Digits);
mrequest.sl = NormalizeDouble(mresult.price + StopLoss,_Digits);
ZeroMemory (check_result);
```

```
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
    if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
    if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
    return;
} else {
    if (OrderSend (mrequest, mresult)) {
        if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
        {
            Print («SL», mrequest.sl, «TP ",mrequest.tp);
        }
        else
        {
            Print («Retcode ",mresult.retcode);
        }
        } else {
            Print («Retcode ",mresult.retcode);
        }
        }
        }
        // -----
    }
    else
    {
        Print («Retcode ",mresult.retcode);
    }
    } else {
        Print («Retcode ",mresult.retcode);
    }
    }
    }
    // -----
-----

}
// -----
-----

if (TradeSignalBuyStop==true&&BuyOpened==true) {
// -----
-----

    if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
        ZeroMemory (mrequest);
        mrequest.action = TRADE_ACTION_DEAL;
        mrequest.symbol = _Symbol;
        mrequest.volume = Lot;
        mrequest.price = NormalizeDouble(latest_price.bid,_Digits);
        mrequest.sl = NormalizeDouble (0.0,_Digits);
```

```
mrequest.tp = NormalizeDouble (0.0, _Digits);
mrequest.deviation=10;
mrequest.type = ORDER_TYPE_SELL;
mrequest.type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
    if(check_result.retcodes==10014) Alert («Неправильный объем в запросе»);
    if(check_result.retcodes==10015) Alert («Неправильная цена в запросе»);
    if(check_result.retcodes==10016) Alert («Неправильные стопы в запросе»);
    if(check_result.retcodes==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
    return;
} else {
    if (OrderSend (mrequest, mresult)) {
        if(mresult.retcodes==10009 || mresult.retcodes==10008) //запрос выполнен или ордер
успешно помещен
        {
            Print («Price», mresult.price);
        }
        else
        {
            if(mresult.retcodes==10004) //Пеквота
            {
                Print («Requote bid ",mresult.bid);
                Print («Requote ask ",mresult.ask);
            } else {
                Print («Retcodes ",mresult.retcodes);
            }
        }
        } else {
            Print («Retcodes ",mresult.retcodes);
        }
    }
}
// -----

if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest.volume = Lot;
    mrequest.type = ORDER_TYPE_SELL;
    mrequest.type_filling = ORDER_FILLING_FOK;

    ZeroMemory (check_result);
```

```
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
    if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
    if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
    return;
} else {
    if (OrderSend (mrequest, mresult)) {
        if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
        {
            Print («Price», mresult.price);
        }
        else
        {
            Print («Retcode ",mresult.retcod);
        }
        } else {
            Print («Retcode ",mresult.retcod);
        }
        }
        }
        // -----

}
// -----

//+-----+
//| Trade function |
//+-----+
void OnTrade ()
{
    static int _deals;
    ulong _ticket=0;

    if (HistorySelect (0,TimeCurrent ()))
    {
        int i=HistoryDealsTotal () -1;

        if (_deals!=i) {
            _deals=i;
        } else {return;}

        if ((_ticket=HistoryDealGetTicket (i))> 0)
        {
            string _comment=HistoryDealGetString (_ticket, DEAL_COMMENT);
```

```
if (StringFind (_comment, «sl», 0) != -1) {
    flagStopLoss = true;
}
}
}
}

bool OnTradeSignalBuy () {

    bool flagBuy = false;

    if (CopyBuffer (handleIMA18, 0, 0, numberBarOpenPosition, MA18Buffer) < 0)
    {
        return false;
    }

    if (CopyBuffer (handleIMA28, 0, 0, numberBarOpenPosition, MA28Buffer) < 0)
    {
        return false;
    }

    if (CopyBuffer (handleIWMA5, 0, 0, numberBarOpenPosition, WMA5Buffer) < 0)
    {
        return false;
    }

    if (CopyBuffer (handleIWMA8, 0, 0, numberBarOpenPosition, WMA8Buffer) < 0)
    {
        return false;
    }

    ArraySetAsSeries (MA18Buffer, true);
    ArraySetAsSeries (MA28Buffer, true);
    ArraySetAsSeries (WMA5Buffer, true);
    ArraySetAsSeries (WMA8Buffer, true);

    bool flagCross1 = false;
    bool flagCross2 = false;
    bool flagCross = false;

    if (WMA5Buffer [1] > MA18Buffer [1] && WMA5Buffer [1] > MA28Buffer [1]
    && WMA8Buffer [1] > MA18Buffer [1] && WMA8Buffer [1] > MA28Buffer [1]) {
        for (int i = 2; i < numberBarOpenPosition; i++) {
            if (WMA5Buffer [i] < MA18Buffer [i] && WMA5Buffer [i] < MA28Buffer [i]) {
                flagCross1 = true;
            }
            if (WMA8Buffer [i] < MA18Buffer [i] && WMA8Buffer [i] < MA28Buffer [i]) {
                flagCross2 = true;
            }
        }
    }
```

```
    }
    if (flagCross1==true&&flagCross2==true) {
        flagCross=true;
    }
}

flagBuy=flagCross;

return flagBuy;
}
// -----
bool OnTradeSignalBuyStop (MqlRates& mrate []) {

    bool flagBuyStop=false;

    if (CopyBuffer (handleIWMA5,0,0,numberBarStopPosition, WMA5Buffer) <0)
    {
        return false;
    }

    if (CopyBuffer (handleIWMA8,0,0,numberBarStopPosition, WMA8Buffer) <0)
    {
        return false;
    }

    ArraySetAsSeries (WMA5Buffer, true);
    ArraySetAsSeries (WMA8Buffer, true);

    bool flagCross=false;

    if (WMA5Buffer [1] <WMA8Buffer [1]) {
        for (int i=2;i <numberBarStopPosition; i++) {
            if (WMA5Buffer [i]> WMA8Buffer [i]) {
                flagCross=true;
            }
        }
    }

    double max=mrate [1].high;

    for (int i=1;i <numberBarStopPosition; i++) {
        if (mrate [i].high> max) max=mrate [i].high;
    }

    if (flagCross==true&&mrate [1].high <=max&&mrate [numberBarStopPosition-1].high
    <=max) {
        flagBuyStop=true;
    }
}
```

```
return flagBuyStop;
}
// -----
bool OnTradeSignalSell () {

bool flagSell=false;

if (CopyBuffer (handleIMA18,0,0,numberBarOpenPosition, MA18Buffer) <0)
{
return false;
}

if (CopyBuffer (handleIMA28,0,0,numberBarOpenPosition, MA28Buffer) <0)
{
return false;
}

if (CopyBuffer (handleIWMA5,0,0,numberBarOpenPosition, WMA5Buffer) <0)
{
return false;
}

if (CopyBuffer (handleIWMA8,0,0,numberBarOpenPosition, WMA8Buffer) <0)
{
return false;
}

ArraySetAsSeries (MA18Buffer, true);
ArraySetAsSeries (MA28Buffer, true);
ArraySetAsSeries (WMA5Buffer, true);
ArraySetAsSeries (WMA8Buffer, true);

bool flagCross1=false;
bool flagCross2=false;
bool flagCross=false;

if (WMA5Buffer [1] <MA18Buffer [1] &&WMA5Buffer [1] <MA28Buffer [1]
&&WMA8Buffer [1] <MA18Buffer [1] &&WMA8Buffer [1] <MA28Buffer [1]) {
for (int i=2;i <numberBarOpenPosition; i++) {
if (WMA5Buffer [i]> MA18Buffer [i] &&WMA5Buffer [i]> MA28Buffer [i]) {
flagCross1=true;
}
if (WMA8Buffer [i]> MA18Buffer [i] &&WMA8Buffer [i]> MA28Buffer [i]) {
flagCross2=true;
}
}
if (flagCross1==true&&flagCross2==true) {
flagCross=true;
}
}
```

```
    }

    flagSell=flagCross;

    return flagSell;
}
// -----
bool OnTradeSignalSellStop (MqlRates& mrate []) {

    bool flagSellStop=false;

    if (CopyBuffer (handleIWMA5,0,0,numberBarStopPosition, WMA5Buffer) <0)
    {
        return false;
    }

    if (CopyBuffer (handleIWMA8,0,0,numberBarStopPosition, WMA8Buffer) <0)
    {
        return false;
    }

    ArraySetAsSeries (WMA5Buffer, true);
    ArraySetAsSeries (WMA8Buffer, true);

    bool flagCross=false;

    if (WMA5Buffer [1]> WMA8Buffer [1]) {
        for (int i=2;i <numberBarStopPosition; i++) {
            if (WMA5Buffer [i] <WMA8Buffer [i]) {
                flagCross=true;
            }
        }
    }

    double min=mrate [1].low;

    for (int i=1;i <numberBarStopPosition; i++) {
        if (mrate [i].low <min) min=mrate [i].low;
    }

    if (flagCross==true&&mrate [1].low> =min&&mrate [numberBarStopPosition-1].low>
=min) {
        flagSellStop=true;
    }

    return flagSellStop;
}
```

Здесь перед функциями обратного вызова, объявляются входные параметры numberBarOpenPosition – количество баров, на которых будет проверяться пересечение

5WMA и 8 WMA туннеля из 18 EMA и 28 EMA, и numberBarStopPosition – количество баров, на которых будет проверяться пересечение 5WMA и 8 WMA и достижения ценой вершины или дна.

Далее объявляются хэндлы индикаторов и динамические массивы значений индикаторов.

В методе OnInit с помощью стандартных функций возвращаются хэндлы индикаторов.

В методе OnTick для получения сигналов на продажу или покупку вызываются функции OnTradeSignalBuy, OnTradeSignalSell, OnTradeSignalBuyStop, OnTradeSignalSellStop.

В функции OnTradeSignalBuy и функции OnTradeSignalSell с помощью хэндлов индикаторов заполняются динамические массивы значений индикаторов и на количестве баров numberBarOpenPosition проверяется пресечение 5WMA и 8 WMA туннеля из 18 EMA и 28 EMA.

В функции OnTradeSignalBuyStop и функции OnTradeSignalSellStop с помощью хэндлов индикаторов заполняются динамические массивы значений индикаторов и на количестве баров numberBarStopPosition проверяется пресечение 5WMA и 8 WMA и достижения ценой вершины или дна.

После компиляции советника в клиентском терминале нажмем правой кнопкой мышки на советнике и выберем Тестировать.

Попробуем оптимизировать параметры numberBarOpenPosition и numberBarStopPosition.

Переменная	Значение	Старт	Шаг	Стоп	Шаги
<input type="checkbox"/> Lot	1.0	1.0	0.1	10.0	
<input type="checkbox"/> EA_Magic	1000	1000	1	10000	
<input type="checkbox"/> spreadLevel	5.0	5.0	0.5	50.0	
<input type="checkbox"/> StopLoss	0.01	0.01	0.001	0.1	
<input type="checkbox"/> Profit	0.01	0.01	0.001	0.1	
<input checked="" type="checkbox"/> numberBarOpenPosition	10	3	1	15	13
<input checked="" type="checkbox"/> numberBarStopPosition	10	3	1	15	13
					169

В результате оптимизации получим профит при значении `numberBarOpenPosition = 3`, независимо от значений `numberBarStopPosition`.

Так получилось, потому что только два раза на трех барах произошло пересечение 5WMA и 8 WMA туннеля из 18 EMA и 28 EMA и оба раза сделки были прибыльными.

Если взять больший период, картина будет не такая радужная, хотя профит сохранится при значении `numberBarOpenPosition = 3`, независимо от значений `numberBarStopPosition`.

При изменении параметра цены индикаторов на PRICE_WEIGHTED показатель прибыли улучшается.

Условие достижения рынком дна или вершины можно заменить на горизонтальность линии EMA.

`MathAbs (WMA5Buffer [1] -WMA5Buffer [numberBarStopPosition-1]) <0.001`

Можно убрать действие сигналов `TradeSignalBuyStop` и `TradeSignalSellStop` и работать только на достижение Take Profit или Stop Loss.

Пример создания эксперта с использованием ООП

В предыдущем примере советника выделим функции OnCheckTradeInit, OnCheckTradeTick, OnTradeSignalBuy, OnTradeSignalBuyStop, OnTradeSignalSell, OnTradeSignalSellStop, а также код открытия и закрытия позиции в отдельные классы.

Проверочные функции OnCheckTradeInit и OnCheckTradeTick выделим в класс CheckTrade.

```
class CheckTrade
{
private:

public:
    CheckTrade ();
    ~CheckTrade ();
    int OnCheckTradeInit (double lot);
    int OnCheckTradeTick (double lot, double spread);
};
//+-----+
//|
//+-----+
CheckTrade::CheckTrade ()
{
}
//+-----+
//|
//+-----+
CheckTrade::~~CheckTrade ()
{
}
//+-----+
int CheckTrade::OnCheckTradeInit (double lot) {
//Проверка запуска эксперта на реальном счете
if ((ENUM_ACCOUNT_TRADE_MODE) AccountInfoInteger
(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL) {
    int mb=MessageBox («Запустить советник на реальном счете?», «Message Box»,
    MB_YESNO|MB_ICONQUESTION);
    if (mb==IDNO) return (0);
}
// -----
//Проверки: запрещена торговля в случае подключения к счету в режиме инвестора,
//отсутствия соединения к серверу, запрета торговли на стороне сервера, если счет
отправлен в архив
//брокер запрещает автоматическую торговлю

if (!TerminalInfoInteger (TERMINAL_CONNECTED)) {
    Alert («No connection to the trade server»);
    return (0);
} else {
```

```
if (!AccountInfoInteger (ACCOUNT_TRADE_ALLOWED)) {
    Alert («Trade for this account is prohibited»);
    return (0);
}

if (!AccountInfoInteger (ACCOUNT_TRADE_EXPERT)) {
    Alert («Trade with the help of experts for the account is prohibited»);
    return (0);
}

// -----

//Проверить корректность объема, с которым мы собираемся выйти на рынок
if (lot <SymbolInfoDouble (Symbol (),SYMBOL_VOLUME_MIN) ||lot>
SymbolInfoDouble (Symbol (),SYMBOL_VOLUME_MAX)) {
    Alert («Lot is not correct!!!»);
    return (0);
}
return (INIT_SUCCEEDED);

}

int CheckTrade::OnCheckTradeTick (double lot, double spread) {

    //Проверка отсутствия соединения к серверу
    if (!TerminalInfoInteger (TERMINAL_CONNECTED)) {
        Alert («No connection to the trade server»);
        return (0);
    }

    //Включена ли кнопка авто-торговли в клиентском терминале
    if (!TerminalInfoInteger (TERMINAL_TRADE_ALLOWED)) {
        Alert («Разрешение на автоматическую торговлю выключено!»);
        return (0);
    }

    //Разрешение на торговлю с помощью эксперта отключено в общих свойствах самого
эксперта
    if (!MQLInfoInteger (MQL_TRADE_ALLOWED)) {
        Alert («Автоматическая торговля запрещена в свойствах эксперта», __FILE__);
        return (0);
    }

    // -----

    //Уровень залоговых средств, при котором требуется пополнение счета
    if ((ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger
(ACCOUNT_MARGIN_SO_MODE) ==ACCOUNT_STOPOUT_MODE_PERCENT) {
```



```
    }
    else
    {
        Print (GetLastError ());
    }

    // -----

    //Контроль над спредом брокера
    double _spread=SymbolInfoInteger (Symbol (),SYMBOL_SPREAD) *MathPow (10, -
SymbolInfoInteger (Symbol (),SYMBOL_DIGITS)) /MathPow (10, -4);

    if (_spread> spread) {
        Alert («Слишком большой спред!»);
        return (0);
    }

    // -----

    //Проверка ограничений на торговые операции по символу, установленные брокером
    if ((ENUM_SYMBOL_TRADE_MODE) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_MODE)!=SYMBOL_TRADE_MODE_FULL) {
        Alert («Установлены ограничения на торговые операции»);
        return (0);
    }

    // -----

    //Достаточно ли баров в истории для расчета советника
    if (Bars (Symbol (), 0) <100)
    {
        Alert («In the chart little bars, Expert will not work!!»);
        return (0);
    }

    return (1);
}

Код открытия и закрытия позиции выделим в класс Trade.
class Trade
{
private:
    double StopLoss;
    double Profit;
    double Lot;
public:
    Trade (double stopLoss, double profit, double lot);
    ~Trade ();
    void Order (bool Buy, bool StopBuy, bool Sell, bool StopSell);
};
```

```
//+-----+
//|
//+-----+
Trade::Trade (double stopLoss, double profit, double lot)
{
StopLoss=stopLoss;
Profit=profit;
Lot=lot;
}
//+-----+
//|
//+-----+
Trade::~~Trade ()
{
}
//+-----+
Trade::Order (bool Buy, bool BuyStop, bool Sell, bool SellStop) {
//Проверка наличия открытой позиции, чтобы не пытаться открыть ее заново
bool BuyOpened=false;
bool SellOpened=false;

if (PositionSelect (_Symbol) ==true)
{
if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_BUY)
{
BuyOpened=true;
}
else if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_SELL)
{
SellOpened=true;
}
}
}
//-----
```

```
-----
MqlTradeRequest mrequest;
MqlTradeCheckResult check_result;
MqlTradeResult mresult;

MqlTick latest_price;
if (!SymbolInfoTick (_Symbol, latest_price))
{
Alert («Ошибка получения последних котировок – ошибка:», GetLastError (),»!!»);
return;
}
if (Buy==true&&BuyOpened==false) {
//-----
```

```
-----
if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
```

```
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_DEAL;
mrequest.symbol = _Symbol;
mrequest.volume = Lot;
mrequest.price = NormalizeDouble(latest_price.ask,_Digits);
mrequest.sl = NormalizeDouble(latest_price.bid - StopLoss,_Digits);
mrequest.tp = NormalizeDouble(latest_price.ask + Profit,_Digits);
mrequest.deviation=10;
mrequest.type = ORDER_TYPE_BUY;
mrequest.type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
if(check_result.retcod==10015) Alert («Неправильная цена в запросе»);
if(check_result.retcod==10016) Alert («Неправильные стопы в запросе»);
if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
return;
} else {
if (OrderSend (mrequest, mresult)) {
if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
{
Print («Price», mresult.price);
}
else
{
if(mresult.retcod==10004) //Реквота
{
Print («Requote bid ",mresult.bid);
Print («Requote ask ",mresult.ask);
} else {
Print («Retcode ",mresult.retcod);
}
}
} else {
Print («Retcode ",mresult.retcod);
}
}
}
// -----
-----
if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE))==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_DEAL;
```

```
mrequest.symbol = _Symbol;
mrequest.volume = Lot;
mrequest.type = ORDER_TYPE_BUY;
mrequest.type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
    if (check_result.retcode==10014) Alert («Неправильный объем в запросе»);
    if (check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
    return;
} else {
    if (OrderSend (mrequest, mresult)) {
        if (mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
        {
            // — — — — —
            ZeroMemory (mrequest);
            mrequest.action = TRADE_ACTION_SLTP;
            mrequest.symbol = _Symbol;
            mrequest.sl = NormalizeDouble(mresult.price - StopLoss,_Digits);
            mrequest.tp = NormalizeDouble(mresult.price + Profit,_Digits);
            ZeroMemory (check_result);
            ZeroMemory (mresult);
            if (!OrderCheck (mrequest, check_result))
            {
                if (check_result.retcode==10015) Alert («Неправильная цена в запросе»);
                if (check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
                return;
            } else {
                if (OrderSend (mrequest, mresult)) {
                    if (mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
                    {
                        Print («SL», mrequest.sl, «TP ",mrequest.tp);
                    }
                    else
                    {
                        Print («Retcode ",mresult.retcode);
                    }
                } else {
                    Print («Retcode ",mresult.retcode);
                }
            }
            // — — — — —
        }
        else
```

```
{
Print («Retcode ",mresult.retcode);
}
} else {
Print («Retcode ",mresult.retcode);
}
}
}
// -----
}
// -----

if (SellStop==true&&SellOpened==true) {
// -----

if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE))==SYMBOL_TRADE_EXECUTION_INSTANT) {
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_DEAL;
mrequest.symbol = _Symbol;
mrequest. volume = Lot;
mrequest.price = NormalizeDouble(latest_price.ask,_Digits);
mrequest.sl = NormalizeDouble (0.0,_Digits);
mrequest.tp = NormalizeDouble (0.0,_Digits);
mrequest.deviation=10;
mrequest. type = ORDER_TYPE_BUY;
mrequest. type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
return;
} else {
if (OrderSend (mrequest, mresult)) {
if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
{
Print («Price», mresult.price);
}
} else
{
if(mresult.retcode==10004) //Пеквота
```

```
{
Print («Requote bid ",mresult.bid);
Print («Requote ask ",mresult.ask);
} else {
Print («Retcode ",mresult.retcode);
}
}
} else {
Print («Retcode ",mresult.retcode);
}
}
}
// -----
-----
if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE))==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_DEAL;
mrequest.symbol = _Symbol;
mrequest. volume = Lot;
mrequest. type = ORDER_TYPE_BUY;
mrequest. type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
return;
} else {
if (OrderSend (mrequest, mresult)) {
if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
{
Print («Price», mresult.price);
}
else
{
Print («Retcode ",mresult.retcode);
}
} else {
Print («Retcode ",mresult.retcode);
}
}
}
// -----
-----
```

```
    }  
    // -----  
-----  
    if (Sell==true&&SellOpened==false) {  
    // -----  
-----  
        if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol  
(_),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {  
            ZeroMemory (mrequest);  
            mrequest.action = TRADE_ACTION_DEAL;  
            mrequest.symbol = _Symbol;  
            mrequest.volume = Lot;  
            mrequest.price = NormalizeDouble(latest_price.bid,_Digits);  
            mrequest.sl = NormalizeDouble(latest_price.ask + StopLoss,_Digits);  
            mrequest.tp = NormalizeDouble(latest_price.bid – Profit,_Digits);  
            mrequest.deviation=10;  
            mrequest.type = ORDER_TYPE_SELL;  
            mrequest.type_filling = ORDER_FILLING_FOK;  
  
            ZeroMemory (check_result);  
            ZeroMemory (mresult);  
            if (!OrderCheck (mrequest, check_result))  
            {  
                if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);  
                if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);  
                if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);  
                if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-  
нения запроса»);  
                return;  
            } else {  
                if (OrderSend (mrequest, mresult)) {  
                    if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер  
успешно помещен  
                    {  
                        Print («Price», mresult.price);  
                    }  
                    else  
                    {  
                        if(mresult.retcode==10004) //Пеквота  
                        {  
                            Print («Requote bid ",mresult.bid);  
                            Print («Requote ask ",mresult.ask);  
                        } else {  
                            Print («Retcode ",mresult.retcode);  
                        }  
                    }  
                } else {  
                    Print («Retcode ",mresult.retcode);  
                }  
            }  
        }  
    }  
}
```

```
    }
    }
    // -----

    if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
    (),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
        ZeroMemory (mrequest);
        mrequest.action = TRADE_ACTION_DEAL;
        mrequest.symbol = _Symbol;
        mrequest.volume = Lot;
        mrequest.type = ORDER_TYPE_SELL;
        mrequest.type_filling = ORDER_FILLING_FOK;

        ZeroMemory (check_result);
        ZeroMemory (mresult);
        if (!OrderCheck (mrequest, check_result))
        {
            if(check_result.retcodes==10014) Alert («Неправильный объем в запросе»);
            if(check_result.retcodes==10019) Alert («Нет достаточных денежных средств для выпол-
            нения запроса»);
            return;
        } else {
            if (OrderSend (mrequest, mresult)) {
                if(mresult.retcodes==10009 || mresult.retcodes==10008) //запрос выполнен или ордер
                успешно помещен
                {
                    // -----
                    ZeroMemory (mrequest);
                    mrequest.action = TRADE_ACTION_SLTP;
                    mrequest.symbol = _Symbol;
                    mrequest.tp = NormalizeDouble(mresult.price - Profit,_Digits);
                    mrequest.sl = NormalizeDouble(mresult.price + StopLoss,_Digits);
                    ZeroMemory (check_result);
                    ZeroMemory (mresult);
                    if (!OrderCheck (mrequest, check_result))
                    {
                        if(check_result.retcodes==10015) Alert («Неправильная цена в запросе»);
                        if(check_result.retcodes==10016) Alert («Неправильные стопы в запросе»);
                        return;
                    } else {
                        if (OrderSend (mrequest, mresult)) {
                            if(mresult.retcodes==10009 || mresult.retcodes==10008) //запрос выполнен или ордер
                            успешно помещен
                            {
                                Print («SL», mrequest.sl, «TP ",mrequest.tp);
                            }
                        } else
                        {
                            Print («Retcode ",mresult.retcodes);
                        }
                    }
                }
            }
        }
    }
}
```

```
    }
    } else {
    Print («Retcode ",mresult.retcod);
    }
    }
    // -----
    }
    else
    {
    Print («Retcode ",mresult.retcod);
    }
    } else {
    Print («Retcode ",mresult.retcod);
    }
    }
    }
    // -----
    -----

    }
    // -----
    -----

    if (BuyStop==true&&BuyOpened==true) {
    // -----
    -----

    if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
    (),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest. volume = Lot;
    mrequest.price = NormalizeDouble(latest_price.bid,_Digits);
    mrequest.sl = NormalizeDouble (0.0,_Digits);
    mrequest.tp = NormalizeDouble (0.0,_Digits);
    mrequest.deviation=10;
    mrequest. type = ORDER_TYPE_SELL;
    mrequest. type_filling = ORDER_FILLING_FOK;

    ZeroMemory (check_result);
    ZeroMemory (mresult);
    if (!OrderCheck (mrequest, check_result))
    {
    if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
    if(check_result.retcod==10015) Alert («Неправильная цена в запросе»);
    if(check_result.retcod==10016) Alert («Неправильные стопы в запросе»);
    if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
    нения запроса»);
    return;
    } else {
```

```
        if (OrderSend (mrequest, mresult)) {
            if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
            {
                Print («Price», mresult.price);
            }
            else
            {
                if(mresult.retcode==10004) //Реквота
                {
                    Print («Requote bid ",mresult.bid);
                    Print («Requote ask ",mresult.ask);
                } else {
                    Print («Retcode ",mresult.retcode);
                }
            }
            } else {
                Print («Retcode ",mresult.retcode);
            }
        }
        // -----

        if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
            ZeroMemory (mrequest);
            mrequest.action = TRADE_ACTION_DEAL;
            mrequest.symbol = _Symbol;
            mrequest.volume = Lot;
            mrequest.type = ORDER_TYPE_SELL;
            mrequest.type_filling = ORDER_FILLING_FOK;

            ZeroMemory (check_result);
            ZeroMemory (mresult);
            if (!OrderCheck (mrequest, check_result))
            {
                if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
                if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
                return;
            } else {
                if (OrderSend (mrequest, mresult)) {
                    if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
                    {
                        Print («Price», mresult.price);
                    }
                    else
                    {

```

```
Print («Retcode ",mresult.retcode);
}
} else {
Print («Retcode ",mresult.retcode);
}
}
}
// -----
```

```
}
// -----
}
Функции сигналов торговой системы выделим в класс Sidus.
class Sidus
{
private:
```

```
int numberBarOpenPosition;
int numberBarStopPosition;
```

```
int handleIMA18;
double MA18Buffer [];
int handleIMA28;
double MA28Buffer [];
int handleIWMA5;
double WMA5Buffer [];
int handleIWMA8;
double WMA8Buffer [];
```

```
public:
Sidus (int BarOpenPosition, int BarStopPosition);
~Sidus ();
bool OnTradeSignalBuy ();
bool OnTradeSignalBuyStop (MqlRates& mrate []);
bool OnTradeSignalSell ();
bool OnTradeSignalSellStop (MqlRates& mrate []);
};
```

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

```
Sidus::Sidus (int BarOpenPosition, int BarStopPosition)
{
numberBarOpenPosition=BarOpenPosition;
numberBarStopPosition=BarStopPosition;
handleIMA18=iMA (_Symbol, PERIOD_H1,18,0,MODE_EMA, PRICE_WEIGHTED);
handleIMA28=iMA (_Symbol, PERIOD_H1,28,0,MODE_EMA, PRICE_WEIGHTED);
handleIWMA5=iMA (_Symbol, PERIOD_H1,5,0,MODE_LWMA, PRICE_WEIGHTED);
handleIWMA8=iMA (_Symbol, PERIOD_H1,8,0,MODE_LWMA, PRICE_WEIGHTED);
```

```
}
//+-----+
//|
//+-----+
Sidus::~Sidus ()
{
}
//+-----+
bool Sidus::OnTradeSignalBuy () {

    bool flagBuy=false;

    if (CopyBuffer (handleIMA18,0,0,numberBarOpenPosition, MA18Buffer) <0)
    {
        return false;
    }

    if (CopyBuffer (handleIMA28,0,0,numberBarOpenPosition, MA28Buffer) <0)
    {
        return false;
    }

    if (CopyBuffer (handleIWMA5,0,0,numberBarOpenPosition, WMA5Buffer) <0)
    {
        return false;
    }

    if (CopyBuffer (handleIWMA8,0,0,numberBarOpenPosition, WMA8Buffer) <0)
    {
        return false;
    }

    ArraySetAsSeries (MA18Buffer, true);
    ArraySetAsSeries (MA28Buffer, true);
    ArraySetAsSeries (WMA5Buffer, true);

    bool flagCross1=false;
    bool flagCross2=false;
    bool flagCross=false;

    if (WMA5Buffer [1]> MA18Buffer [1] &&WMA5Buffer [1]> MA28Buffer [1]
    &&WMA8Buffer [1]> MA18Buffer [1] &&WMA8Buffer [1]> MA28Buffer [1]) {
        for (int i=2;i <numberBarOpenPosition; i++) {
            if (WMA5Buffer [i] <MA18Buffer [i] &&WMA5Buffer [i] <MA28Buffer [i]) {
                flagCross1=true;
            }
            if (WMA8Buffer [i] <MA18Buffer [i] &&WMA8Buffer [i] <MA28Buffer [i]) {
                flagCross2=true;
            }
        }
    }
```

```
    }
    if (flagCross1==true&&flagCross2==true) {
        flagCross=true;
    }
}

if (flagCross==true) {
    flagBuy=true;
}

return flagBuy;
}

bool Sidus::OnTradeSignalBuyStop (MqlRates& mrate []) {

    bool flagBuyStop=false;

    if (CopyBuffer (handleIWMA5,0,0,numberBarStopPosition, WMA5Buffer) <0)
    {
        return false;
    }

    if (CopyBuffer (handleIWMA8,0,0,numberBarStopPosition, WMA8Buffer) <0)
    {
        return false;
    }

    ArraySetAsSeries (WMA5Buffer, true);
    ArraySetAsSeries (WMA8Buffer, true);

    bool flagCross=false;

    if (WMA5Buffer [1] <WMA8Buffer [1]) {
        for (int i=2;i <numberBarStopPosition; i++) {
            if (WMA5Buffer [i]> WMA8Buffer [i]) {
                flagCross=true;
            }
        }
    }

    double max=mrate [1].high;

    for (int i=1;i <numberBarStopPosition; i++) {
        if (mrate [i].high> max) max=mrate [i].high;
    }

    if (flagCross==true&&mrate [1].high <=max&&mrate [numberBarStopPosition-1].high
    <=max) {
        flagBuyStop=true;
    }
}
```

```
    }

    return flagBuyStop;
}

bool Sidus::OnTradeSignalSell () {

    bool flagSell=false;

    if (CopyBuffer (handleIMA18,0,0,numberBarOpenPosition, MA18Buffer) <0)
    {
        return false;
    }

    if (CopyBuffer (handleIMA28,0,0,numberBarOpenPosition, MA28Buffer) <0)
    {
        return false;
    }

    if (CopyBuffer (handleIWMA5,0,0,numberBarOpenPosition, WMA5Buffer) <0)
    {
        return false;
    }

    if (CopyBuffer (handleIWMA8,0,0,numberBarOpenPosition, WMA8Buffer) <0)
    {
        return false;
    }

    ArraySetAsSeries (MA18Buffer, true);
    ArraySetAsSeries (MA28Buffer, true);
    ArraySetAsSeries (WMA5Buffer, true);
    ArraySetAsSeries (WMA8Buffer, true);

    bool flagCross1=false;
    bool flagCross2=false;
    bool flagCross=false;

    if (WMA5Buffer [1] <MA18Buffer [1] &&WMA5Buffer [1] <MA28Buffer [1]
    &&WMA8Buffer [1] <MA18Buffer [1] &&WMA8Buffer [1] <MA28Buffer [1]) {
        for (int i=2;i <numberBarOpenPosition; i++) {
            if (WMA5Buffer [i]> MA18Buffer [i] &&WMA5Buffer [i]> MA28Buffer [i]) {
                flagCross1=true;
            }
            if (WMA8Buffer [i]> MA18Buffer [i] &&WMA8Buffer [i]> MA28Buffer [i]) {
                flagCross2=true;
            }
        }
    }
    if (flagCross1==true&&flagCross2==true) {
```

```
    flagCross=true;
  }
}

if (flagCross==true) {
  flagSell=true;
}
return flagSell;
}

bool Sidus::OnTradeSignalSellStop (MqlRates& mrate []) {

  bool flagSellStop=false;

  if (CopyBuffer (handleIWMA5,0,0,numberBarStopPosition, WMA5Buffer) <0)
  {
    return false;
  }

  if (CopyBuffer (handleIWMA8,0,0,numberBarStopPosition, WMA8Buffer) <0)
  {
    return false;
  }

  ArraySetAsSeries (WMA5Buffer, true);
  ArraySetAsSeries (WMA8Buffer, true);

  bool flagCross=false;

  if (WMA5Buffer [1]> WMA8Buffer [1]) {
    for (int i=2;i <numberBarStopPosition; i++) {
      if (WMA5Buffer [i] <WMA8Buffer [i]) {
        flagCross=true;
      }
    }
  }

  double min=mrate [1].low;

  for (int i=1;i <numberBarStopPosition; i++) {
    if (mrate [i].low <min) min=mrate [i].low;
  }

  if (flagCross==true&&mrate [1].low> =min&&mrate [numberBarStopPosition-1].low>
=min) {
    flagSellStop=true;
  }

  return flagSellStop;
}
```

```
}
```

В результате выделения функций в отдельные классы код советника существенно сократится.

```
#include "CheckTrade.mqh»
```

```
#include "Trade.mqh»
```

```
#include "Sidus.mqh»
```

```
input double Lot=1;
```

```
input int EA_Magic=1000;
```

```
input double spreadLevel=5.0;
```

```
input double StopLoss=0.01;
```

```
input double Profit=0.01;
```

```
input int numberBarOpenPosition=3;
```

```
input int numberBarStopPosition=5;
```

```
CheckTrade checkTrade;
```

```
Trade trade (StopLoss, Profit, Lot);
```

```
Sidus sidus (numberBarOpenPosition, numberBarStopPosition);
```

```
bool flagStopLoss=false;
```

```
//+-----+
//| Expert initialization function |
//+-----+
int OnInit ()
```

```
{
return (checkTrade. OnCheckTradeInit (Lot));
}
```

```
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit (const int reason)
```

```
{
}
//+-----+
//| Expert tick function |
//+-----+
void OnTick ()
```

```
{
if (!checkTrade. OnCheckTradeTick (Lot, spreadLevel)) {
return;
}
```

```

}

//Ограничить вычисления советника по появлению нового бара на графике
static datetime last_time;
```

```
datetime last_bar_time= (datetime) SeriesInfoInteger (Symbol (),Period
(),SERIES_LASTBAR_DATE);
if (last_time!=last_bar_time)
```

```
{
last_time=last_bar_time;
} else {
return;
}

//Ограничить вычисления советника по flagStopLoss
static datetime last_time_daily;
datetime last_bar_time_daily= (datetime) SeriesInfoInteger (Symbol
(),PERIOD_D1,SERIES_LASTBAR_DATE);
if (last_time_daily!=last_bar_time_daily)
{
last_time_daily=last_bar_time_daily;
flagStopLoss=false;
}

if (flagStopLoss==true) return;

//Для вычисления сигналов торговой системы требуются исторические данные сим-
вола

MqlRates mrate [];
ResetLastError ();
if (CopyRates (Symbol (),Period (),0,numberBarStopPosition, mrate) <0)
{
Print (GetLastError ());
return;
}

ArraySetAsSeries (mrate, true);

// -----

bool TradeSignalBuy=false;
bool TradeSignalSell=false;

TradeSignalBuy=sidus. OnTradeSignalBuy ();
TradeSignalSell=sidus. OnTradeSignalSell ();

bool TradeSignalBuyStop=false;
bool TradeSignalSellStop=false;

TradeSignalBuyStop=sidus. OnTradeSignalBuyStop (mrate);
TradeSignalSellStop=sidus. OnTradeSignalSellStop (mrate);

trade. Order (TradeSignalBuy, TradeSignalBuyStop, TradeSignalSell, TradeSignalSellStop);
}
```

```
//+-----+
//| Trade function |
//+-----+
void OnTrade ()
{
    static int _deals;
    ulong _ticket=0;

    if (HistorySelect (0,TimeCurrent ()))
    {
        int i=HistoryDealsTotal () -1;

        if (_deals!=i) {
            _deals=i;
        } else {return;}

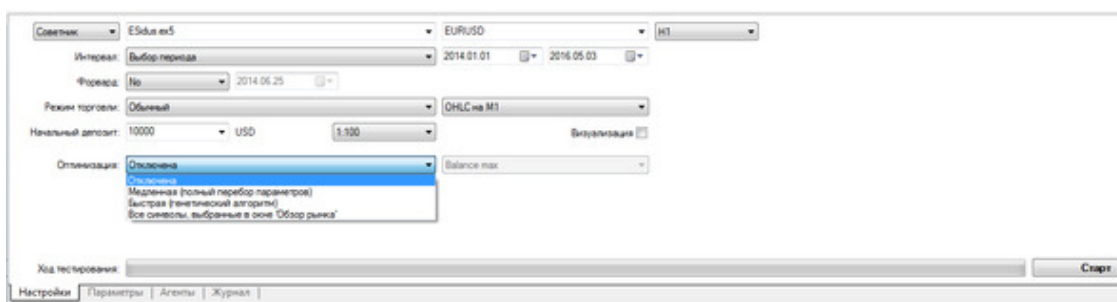
        if ((_ticket=HistoryDealGetTicket (i))> 0)
        {
            string _comment=HistoryDealGetString (_ticket, DEAL_COMMENT);
            if (StringFind (_comment,«sl», 0)!=-1) {
                flagStopLoss=true;
            }
        }
    }
}

//-----
```

Тестирование советников

Встроенный тестер терминала MetaTrader 5 позволяет протестировать и оптимизировать входные input параметры советника с использованием исторических данных финансовых инструментов.

Открыть тестер можно, нажав правой кнопкой мышки на советнике в окне Навигатор терминала и в контекстном меню выбрав Тестировать.



Также тестер можно открыть в меню Вид терминала с помощью команды Тестер стратегий.

Тестирование является многопоточным и проводится с помощью специальных сервис-агентов, которые представлены тремя типами.

Это локальные агенты, устанавливаемые вместе с клиентским терминалом. Число локальных агентов равно числу логических ядер процессора компьютера, которые используются агентами параллельно, поэтому при тестировании задействуются все доступные ресурсы компьютера.

Также для тестирования можно использовать локальные сетевые агенты, установленные на других компьютерах вместе с клиентскими терминалами. Компьютеры должны быть соединены в локальную сеть, после чего создается ферма агентов с помощью менеджера агентов в меню Сервис терминала и производится подключение агентов во вкладке Агенты тестера терминала.

Локальные сетевые агенты также могут быть установлены на компьютере отдельно от торговой платформы с помощью файла metatester64.exe. Локальные сетевые агенты можно устанавливать только в 64-битных системах.

Еще один тип агентов это облачные агенты, использование которых платное и подключить которые также можно во вкладке Агенты тестера терминала.

Окно тестера позволяет выбрать интервал тестирования, финансовый инструмент для тестирования, если он явно не задан в советнике, временной период финансового инструмента, включить оптимизацию входных параметров советника, визуальный режим тестирования и др.

При включении визуального режима тестирования на графике показываются используемые советником индикаторы.

Вкладка Бэктест окна тестера показывает результаты тестирования советника.

Для максимальной приближенности к условиям реальной торговли, при тестировании можно выбрать режим «Произвольная задержка», «Каждый тик на основе реальных тиков».

Быстро провести оптимизацию входных параметров можно в режиме «Форвард: 1/4», «Обычный», «Только цены открытия».

Для того чтобы оценить эффективность советника создадим индикатор, показывающий все потенциальные сделки на покупку и продажу инструмента.

Данный индикатор будет основываться на скользящей средней. При возрастании значения скользящей средней будет рассчитываться позиция на покупку, при убывании значения скользящей средней будет идти расчет позиции на продажу.

```
#property indicator_chart_window
```

```
#property indicator_buffers 2
```

```
#property indicator_plots 1
```

```
#property indicator_type1 DRAW_COLOR_LINE
```

```
#property indicator_color1 clrBlack, clrRed, clrBlueViolet
```

```
#property indicator_style1 STYLE_SOLID
```

```
#property indicator_width1 2
```

```
input int ma_period=5;
```

```
input double delta=0.0001;
```

```
input int shift_delta=1;
```

```
int ma_shift=0; // смещение
```

```
ENUM_MA_METHOD ma_method=MODE_EMA; // тип сглаживания
```

```
ENUM_APPLIED_PRICE applied_price=PRICE_WEIGHTED; // тип цены
```

```
string symbol=_Symbol; // символ
```

```
ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
```

```
double InBuffer [];
```

```
double ColorBuffer [];
```

```
int handleMA;
```

```
// - - переменная для хранения
```

```
string name=symbol;
```

```
// - - имя индикатора на графике
```

```
string short_name;
```

```
// - - будем хранить количество значений в индикаторе Average Directional Movement
```

Index

```
int bars_calculated=0;
```

```
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit ()
```

```
{
```

```
// - - indicator buffers mapping
```

```
SetIndexBuffer (0,InBuffer, INDICATOR_DATA);
```

```
SetIndexBuffer (1,ColorBuffer, INDICATOR_COLOR_INDEX);
```

```
// - - определимся с символом, на котором строится индикатор
```

```
name=symbol;
```

```
// - удалим пробелы слева и справа
StringTrimRight (name);
StringTrimLeft (name);
// - если после этого длина строки name нулевая
if (StringLen (name) ==0)
{
// - возьмем символ с графика, на котором запущен индикатор
name=_Symbol;
}
handleMA=iMA (name, period, ma_period, ma_shift, ma_method, applied_price);
short_name=«Profit»;
IndicatorSetString (INDICATOR_SHORTNAME, short_name);
// - —
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 [])
{
// - количество копируемых значений из индикатора
int values_to_copy;
// - узнаем количество рассчитанных значений в индикаторе
int calculated=BarsCalculated (handleMA);
if (calculated <=0)
{
PrintFormat («BarsCalculated () вернул %d, код ошибки %d», calculated, GetLastError ());
return (0);
}
// - если это первый запуск вычислений нашего индикатора или изменилось количество значений в индикаторе
// - или если необходимо рассчитать индикатор для двух или более баров (значит что-то изменилось в истории)
if (prev_calculated==0 || calculated!=bars_calculated || rates_total> prev_calculated+1)
{
// - если массив больше, чем значений в индикаторе на паре symbol/period, то копируем не все
// - в противном случае копировать будем меньше, чем размер индикаторных буферов
if (calculated> rates_total) values_to_copy=rates_total;
else values_to_copy=calculated;
```

```
    }
    else
    {
        // - - значит наш индикатор рассчитывается не в первый раз и с момента последнего
        вызова OnCalculate ()
        // - - для расчета добавилось не более одного бара
        values_to_copy= (rates_total-prev_calculated) +1;
    }

    // - - если FillArraysFromBuffer вернула false, значит данные не готовы – завершаем
    работу
    if (!FillArrayFromBufferMA (InBuffer, ma_shift, handleMA, values_to_copy)) return (0);

    // - - заппомним количество значений в индикаторе Average Directional Movement Index

    bars_calculated=calculated;
    if (values_to_copy> 1)
    {
        bool flagSell=false;
        double priceSellOpen;
        double priceSellStop;
        bool flagBuy=false;
        double priceBuyOpen;
        double priceBuyStop;

        int size=values_to_copy;
        for (int i=shift_delta; i < (size-1); i++) {

            ColorBuffer [i] =0;

            if ((InBuffer [i-shift_delta] -InBuffer [i])> delta) {
                ColorBuffer [i] =1;
                if (flagSell==false) {
                    priceSellOpen=open [i];
                    if (!ObjectCreate (0,«Sell»+i, OBJ_ARROW,0,time [i],low [i]))
                    {
                        return (false);
                    }
                    ObjectSetInteger (0,«Sell»+i, OBJPROP_COLOR, clrRed);
                    ObjectSetInteger (0,«Sell»+i, OBJPROP_ARROWCODE,234);
                    ObjectSetInteger (0,«Sell»+i, OBJPROP_WIDTH,1);
                    ObjectSetInteger (0,«Sell»+i, OBJPROP_ANCHOR, ANCHOR_LOWER);
                    ObjectSetInteger (0,«Sell»+i, OBJPROP_HIDDEN, true);
                    ObjectSetString (0,«Sell»+i, OBJPROP_TOOLTIP,»»»+close [i]);

                }
                flagSell=true;
            } else {
                if (flagSell==true) {
```

```
priceSellStop=open [i];
double profit=priceSellOpen-priceSellStop;
if (profit> spread [i] /MathPow (10,Digits ())) {
if (!ObjectCreate (0,«SellStop»+i, OBJ_TEXT,0,time [i],low [i]))
{
return (false);
}
ObjectSetString (0,«SellStop»+i, OBJPROP_TEXT,«Profit: "+DoubleToString
(profit*MathPow (10,Digits ()),1));
ObjectSetDouble (0,«SellStop»+i, OBJPROP_ANGLE,90.0);
ObjectSetInteger (0,«SellStop»+i, OBJPROP_COLOR, clrRed);
}
flagSell=false;
}
}

if ((InBuffer [i] -InBuffer [i-shift_delta])> delta) {
ColorBuffer [i] =2;
if (flagBuy==false) {
priceBuyOpen=open [i];
if (!ObjectCreate (0,«Buy»+i, OBJ_ARROW,0,time [i],high [i]))
{
return (false);
}

ObjectSetInteger (0,«Buy»+i, OBJPROP_COLOR, clrGreen);
ObjectSetInteger (0,«Buy»+i, OBJPROP_ARROWCODE,233);
ObjectSetInteger (0,«Buy»+i, OBJPROP_WIDTH,1);
ObjectSetInteger (0,«Buy»+i, OBJPROP_ANCHOR, ANCHOR_UPPER);
ObjectSetInteger (0,«Buy»+i, OBJPROP_HIDDEN, true);
ObjectSetString (0,«Buy»+i, OBJPROP_TOOLTIP,»»+close [i]);
}
flagBuy=true;
} else {
if (flagBuy==true) {
priceBuyStop=open [i];
double profit=priceBuyStop-priceBuyOpen;
if (profit> spread [i] /MathPow (10,Digits ())) {
if (!ObjectCreate (0,«BuyStop»+i, OBJ_TEXT,0,time [i],high [i]))
{
return (false);
}
ObjectSetString (0,«BuyStop»+i, OBJPROP_TEXT,«Profit: "+DoubleToString
(profit*MathPow (10,Digits ()),1));
ObjectSetDouble (0,«BuyStop»+i, OBJPROP_ANGLE,90.0);
ObjectSetInteger (0,«BuyStop»+i, OBJPROP_COLOR, clrBlueViolet);
}
flagBuy=false;
}
}
```

```
    }

    }
}

return (rates_total);
}
//+-----+
//| Заполняем индикаторный буфер из индикатора iMA |
//+-----+
bool FillArrayFromBufferMA (double &values [], // индикаторный буфер значений
Moving Average
int shift, // смещение
int ind_handle, // хэндл индикатора iMA
int amount // количество копируемых значений
)
{
    // - сбросим код ошибки
    ResetLastError ();
    // - заполняем часть массива iMABuffer значениями из индикаторного буфера под
индексом 0
    if (CopyBuffer (ind_handle,0, -shift, amount, values) <0)
    {
        // - если копирование не удалось, сообщим код ошибки
        PrintFormat («Не удалось скопировать данные из индикатора iMA, код ошибки %d»,
GetLastError ());
        // - завершим с нулевым результатом – это означает, что индикатор будет считаться
нерассчитанным
        return (false);
    }
    // - все получилось
    return (true);
}

void OnDeinit (const int reason) {
    ObjectsDeleteAll (0, -1, -1);
}
```



Присоединим данный индикатор к советнику, рассмотренному в предыдущем примере.

```
int handleProfit;
double ProfitBuffer [];
int OnInit ()
{

    handleProfit=iCustom (_Symbol, PERIOD_H1,«Profit», 5,0.0001,1);
    return (checkTrade. OnCheckTradeInit (Lot));
}
```

В функции OnTrade советника будем показывать фактически полученный профит от сделки.

```
void OnTrade ()
{
    static int _deals;
    ulong _ticket=0;

    if (HistorySelect (0,TimeCurrent ()))
    {
        int i=HistoryDealsTotal () -1;

        if (_deals!=i) {
            _deals=i;
        } else {return;}

        if ((_ticket=HistoryDealGetTicket (i))> 0)
        {
            string _comment=HistoryDealGetString (_ticket, DEAL_COMMENT);
            if (StringFind (_comment,«sl», 0)!=-1) {
                flagStopLoss=true;
            }
            if (HistoryDealGetDouble (_ticket, DEAL_PROFIT)!=0.0) {
```

```
if (!ObjectCreate (0,«Deal»+_ticket, OBJ_TEXT,0, HistoryDealGetInteger (_ticket,
DEAL_TIME),HistoryDealGetDouble (_ticket, DEAL_PRICE)))
{
return;
}
ObjectSetString (0,«Deal»+_ticket, OBJPROP_TEXT,«Profit: "+HistoryDealGetDouble
(_ticket, DEAL_PROFIT));
ObjectSetDouble (0,«Deal»+_ticket, OBJPROP_ANGLE,90.0);
ObjectSetInteger (0,«Deal»+_ticket, OBJPROP_COLOR, clrYellow);
}
}
}
}
```

Произведем визуальное тестирование советника.



Из визуального тестирования видно, что у данного советника есть проблемы с входом и выходом из рынка, а также много пропущенных потенциальных сделок.

Управление капиталом и оценка эффективности эксперта

Управление капиталом – это способ принятия решения о том, какой частью счета следует рисковать в отдельной торговой возможности.

Типичные правила управлением капиталом:

Соотношение возможных потерь и прибылей 1:3.

Закрывать убыточные позиции раньше, чем прибыльные.

Избегать очень кратковременных позиций.

Не принимать решений перед закрытием торгов.

Общая сумма средств, вкладываемых в одну сделку, не может превышать 10% – 15% от общего капитала.

Общий процент вложенных средств по всем открытым позициям не может превышать 30%.

Максимально допустимый риск на сделку 2—5% от размера депозита.

Общий максимально допустимый риск по всем открытым позициям 5—7% от размера депозита.

Стратегии управления капиталом:

Фиксированная сумма или фиксированный процент капитала, подвергаемый риску при следующей сделке.

Согласование выигрышей и проигрышей при торговле. По этой методике трейдеры определяют объем торговли после успешных выигрышей или проигрышей. Например, после проигранной сделки, они могут решить удвоить объем торговли после следующего сигнала к торговле, чтобы возместить убытки. В этой системе используется статистический анализ для того, чтобы установить взаимосвязь между проигрышем и выигрышем, когда проигрыши и выигрыши или чередуются, или после определенного количества проигрышей следуют выигрыши. При этом можно после выигрыша увеличивать размер открываемых позиций и уменьшать после проигрыша, или после череды проигрышей увеличивать объем позиции, ожидая скорого выигрыша, либо, наоборот, уменьшать, ожидая проигрыша.

Пересечение кривых цены – по этой системе производится анализ торговой системы с помощью длинной и короткой скользящих средних кривых (например, 3 и 8) прибылей и убытков сделок. Если короткая скользящая средняя кривая находится над более длинной скользящей кривой, тогда торговая система дает лучшие результаты, если же короткая скользящая средняя кривая находится под длинной скользящей кривой, тогда система работает хуже и сигналы торговой системы на открытие позиций нужно пропускать. При этом для расчета кривых используются все сигналы торговой системы, а не только те, которые были фактически реализованы. Наличие выигрышных и проигрышных периодов торговой системы также определяется с помощью статистического анализа.

Optimal f и Secure f это определение на основе тестирования торговой системы на прошлых сделках некоторой оптимальной доли начального капитала, инвестируемой в каждую сделку, с целью достижения максимальной доходности данной системы в качестве основного критерия (Optimal f) или достижения максимальной доходности с учётом ограничения предельно допустимых убытков (Secure f).

Зависимость между проигрышами и выигрышами анализируется с помощью Z-счета. Доверительный интервал при этом должен превышать 94%.

Z-счет рассчитывается путем сравнения количества серий в наборе сделок относительно количества серий, которое можно было бы ожидать при статистической независимости результатов текущей сделки от прошедших сделок.

$$Z = (N * (R - 0.5) - X) / ((X * (X - N)) / ((N - 1)) ^ (1/2))$$

Где:

N = общее количество сделок.

X = 2 * (общее число прибыльных сделок) * (общее число убыточных сделок).

R = количество серий в наблюдении. Каждый раз, когда после прибыльной сделки следует убыточная сделка или наоборот считается за серию.

Положительный Z-счет означает, что в наблюдаемой истории сделок количество серий меньше, чем следовало бы ожидать при статистически независимом распределении исходов сделок. Следовательно, есть тенденция к тому, чтобы после выигрыша следовал проигрыш, а после проигрыша – выигрыш. Наличие положительного Z-счета означает, что после убыточной сделки можно по следующему сигналу увеличить размер открываемой позиции, это позволит покрыть убытки и увеличить общую прибыльность системы.

Отрицательный Z-счет означает, что в наблюдаемой истории сделок количество серий больше, чем следовало бы ожидать при статистически независимом распределении исходов сделок. Следовательно, за выигрышной сделкой обычно следует другая выигрышная сделка, а за убыточной сделкой следует другая убыточная сделка.

Наличие отрицательного Z-счета означает, что можно использовать пересечение кривых доходности, при том торговая система будет работать во время прибыльной фазы и выключаться после наступления убыточной фазы.

Значение Z-счета эксперта можно посмотреть во вкладке Бэктест тестера клиентского терминала после тестирования эксперта.

В рассмотренном примере эксперта на основе торговой системы Сидуса, при значениях numberBarOpenPosition=5 и numberBarStopPosition=5, Z-счет эксперта равен 2.89 (99.61%). Однако на этом основании скорректировать работу эксперта достаточно проблематично, так как после выигрыша может следовать не единичный проигрыш, а серия убытков и наоборот.

Вывести последовательность выигрышей и проигрышей эксперта можно в его функции OnDeinit.

```
void OnDeinit (const int reason)
{
    if (HistorySelect (0,TimeCurrent ()))
    {
        int n=HistoryDealsTotal ();
        for (int i=0;i <n;i++) {
            if (HistoryDealGetDouble (HistoryDealGetTicket (i), DEAL_PROFIT)!=0.0) {
                Print («Profit», HistoryDealGetDouble (HistoryDealGetTicket (i), DEAL_PROFIT));
            }
        }
    }
}
```

Общая эффективность советника определяется как разница между реализованной в ходе трейда прибылью и потенциально возможной прибылью за время этого же трейда.

Общая эффективность =
(Реализованная разница цен) / (Потенциальная прибыль)

Для длинных позиций:

Общая эффективность = (Цена закрытия – Цена открытия) /

(Максимальная цена – Минимальная цена)

Для коротких позиций:

Общая эффективность = (Цена открытия – Цена закрытия) /
(Максимальная цена – Минимальная цена)

Эффективность входа в рынок определяется как максимальная разница в ценах относительно цены входа как часть общего потенциала доходности в ходе трейда. Эффективность входа в рынок показывает, насколько хорошо торговая система открывает трейды – в случае длинных позиций это то, насколько сигнал входа близок к наименьшей точке в ходе трейда, в случае коротких позиций – насколько сигнал входа близок к максимальной точке в ходе трейда.

Эффективность входа = (максимальная разница в ценах относительно цены входа) /
(Потенциальная прибыль)

Максимальная разница в ценах относительно цены входа – это разница между максимальной ценой и ценой входа (для коротких позиций – минимальной ценой).

Для длинных позиций:

Эффективность входа = (Максимальная цена – Цена открытия трейда) / (Максимальная цена – Минимальная цена)

Для коротких позиций:

Эффективность входа = (Цена открытия трейда – Минимальная цена) / (Максимальная цена – Минимальная цена)

Эффективность выхода из рынка определяется как максимальная разница в ценах относительно цены выхода как часть общего потенциала доходности в ходе трейда. Эффективность выхода из рынка показывает, насколько хорошо система закрывает трейды – в случае длинных позиций это то, насколько сигнал выхода близок к максимальной точке в ходе трейда, в случае коротких позиций – насколько сигнал выхода близок к минимальной точке в ходе трейда.

Эффективность выхода = (максимальная разница в ценах относительно цены выхода) /
(Потенциальная прибыль)

Максимальная разница в ценах относительно цены выхода – это разница между ценой выхода и минимальной ценой (для коротких позиций – максимальной ценой).

Для длинных позиций:

Эффективность выхода = (Цена выхода – Минимальная цена) / (Максимальная цена – Минимальная цена)

Для коротких позиций:

Эффективность выхода = (Максимальная цена – Цена выхода) / (Максимальная цена – Минимальная цена)

Общая эффективность является суммой эффективности входа и эффективности выхода минус 1.

Эффективность входа и эффективность выхода могут быть позитивными величинами, однако общая эффективность может быть отрицательной величиной. Если сумма эффективности входа и эффективности выхода меньше 100%, то сделка убыточна.

Для анализа работы эксперта вычисляются средняя общая эффективность, средняя эффективность входов и средняя эффективность выходов.

Для вычисления средних величин лучше пользоваться статистическим анализом, чтобы отбрасывать высказывающие сделки.

Торговая система является статистически прибыльной, если:

(Средняя выигрышная сделка) * (% выигрышей) – накладные расходы >

(Средняя проигрышная сделка) * (% проигрышей)

Накладные расходы это проскальзывания, комиссионные и др.

Такие показатели как Средняя выигрышная сделка (Средний прибыльный трейд), % выигрышей (Прибыльные трейды (% от всех)), Средняя проигрышная сделка (Средний убыточный трейд), % проигрышей (Убыточные трейды (% от всех)) можно посмотреть во вкладке Бэктест тестера клиентского терминала после тестирования эксперта.

Показатель статистической прибыльности эксперта без учета накладных расходов или матожидание выигрыша также можно посмотреть во вкладке Бэктест тестера клиентского терминала после тестирования эксперта.

Матожидание выигрыша рассчитывается по следующей формуле.

$$\text{Expected Payoff} = (\text{ProfitTrades} / \text{TotalTrades}) * (\text{GrossProfit} / \text{ProfitTrades}) - (\text{LossTrades} / \text{TotalTrades}) * (\text{GrossLoss} / \text{LossTrades})$$

где:

TotalTrades – общее количество сделок;

ProfitTrades – количество прибыльных сделок;

LossTrades – количество убыточных сделок;

GrossProfit – общая прибыль;

GrossLoss – общий убыток.

Здесь ProfitTrades / TotalTrades это % выигрышей, GrossProfit / ProfitTrades – Средняя выигрышная сделка, LossTrades / TotalTrades – % проигрышей, GrossLoss / LossTrades – Средняя проигрышная сделка.

В рассмотренном примере эксперта на основе торговой системы Сидуса, при значении numberBarOpenPosition=3, матожидание выигрыша составляет 232.62 – 147.55=85. Это меньше 10 пунктов, поэтому можно сказать, что прибыльность эксперта является неустойчивой.

В рассмотренном примере эксперта на основе торговой системы Сидуса заменим сигналы закрытия позиции на использование Trailing Stop, т.е. при достижении цены уровня безубыточности будет передвигать StopLoss.

Для этого изменим класс Trade.

```
class Trade
{
private:
double StopLoss;
double Profit;
double Lot;
double TrailingStop;
double priceBuy;
double priceSell;
double slBuy;
double slSell;

public:
Trade (double stopLoss, double profit, double lot, double trailingStop);
~Trade ();
void Order (bool Buy, bool StopBuy, bool Sell, bool StopSell);
void Trailing ();
};
//+-----+
//|
//+-----+
Trade::Trade (double stopLoss, double profit, double lot, double trailingStop)
```

```

{
StopLoss=stopLoss;
Profit=profit;
Lot=lot;
TrailingStop=trailingStop;
priceBuy=0.0;
priceSell=0.0;
slBuy=0.0;
slSell=0.0;
}
//+-----+
//|
//+-----+
Trade::~Trade ()
{
}
//+-----+
void Trade::Trailing () {
//Проверка наличия открытой позиции, чтобы не пытаться открыть ее заново
bool BuyOpened=false;
bool SellOpened=false;

if (PositionSelect (_Symbol) ==true)
{
if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_BUY)
{
BuyOpened=true;
}
else if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_SELL)
{
SellOpened=true;
}
}
}
// -----
}

MqlTradeRequest mrequest;
MqlTradeCheckResult check_result;
MqlTradeResult mresult;

MqlTick latest_price;
if (!SymbolInfoTick (_Symbol, latest_price))
{
Alert («Ошибка получения последних котировок – ошибка:», GetLastError (),»!!»);
return;
}
if (BuyOpened==true) {
double TBS=0;
if (TrailingStop <SymbolInfoInteger (Symbol (),SYMBOL_TRADE_STOPS_LEVEL)
*_Point) {

```

```
TBS=SymbolInfoInteger (Symbol (),SYMBOL_TRADE_STOPS_LEVEL) * _Point;
} else {
TBS=TrailingStop;
}
if (
latest_price.bid-priceBuy> =TBS
) {
mrequest.action = TRADE_ACTION_SLTP;
mrequest.price = NormalizeDouble(latest_price.ask,_Digits);
mrequest.sl = NormalizeDouble (priceBuy,_Digits); // Stop Loss
mrequest.tp = NormalizeDouble(latest_price.ask + Profit,_Digits);
mrequest.symbol = _Symbol; // символ
mrequest. volume = Lot; // количество лотов для торговли
mrequest. type_filling = ORDER_FILLING_FOK;
mrequest. type = ORDER_TYPE_BUY; // ордер на покупку
// - - отсылаем ордер
if (!OrderCheck (mrequest, check_result))
{
return;
} else {
if (OrderSend (mrequest, mresult)) {}
}

// анализируем код возврата торгового сервера
if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
{
slBuy= mrequest.sl;
priceBuy=slBuy+TrailingStop;
}
else
{
return;
}
}

if (SellOpened==true) {
double TSS=0;
if (TrailingStop <SymbolInfoInteger (Symbol (),SYMBOL_TRADE_STOPS_LEVEL)
*_Point) {
TSS=SymbolInfoInteger (Symbol (),SYMBOL_TRADE_STOPS_LEVEL) * _Point;
} else {
TSS=TrailingStop;
}
if (
priceSell-latest_price.ask> =TSS
) {
mrequest.action = TRADE_ACTION_SLTP;
```

```
mrequest.price = NormalizeDouble(latest_price.bid,_Digits);
mrequest.sl = NormalizeDouble (priceSell,_Digits); // Stop Loss
mrequest.tp = NormalizeDouble(latest_price.bid – Profit,_Digits);
mrequest.symbol = _Symbol; // символ
mrequest. volume = Lot; // количество лотов для торговли
mrequest. type_filling = ORDER_FILLING_FOK;
mrequest. type = ORDER_TYPE_SELL; // ордер на покупку
// – - отсылаем ордер
if (!OrderCheck (mrequest, check_result))
{
return;
} else {
if (OrderSend (mrequest, mresult)) {}
}

// анализируем код возврата торгового сервера
if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
{
slSell= mrequest.sl;
priceSell=slSell-TrailingStop;
}
else
{
return;
}
}

}

//+-----+
void Trade::Order (bool Buy, bool BuyStop, bool Sell, bool SellStop) {
//Проверка наличия открытой позиции, чтобы не пытаться открыть ее заново
bool BuyOpened=false;
bool SellOpened=false;

if (PositionSelect (_Symbol) ==true)
{
if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_BUY)
{
BuyOpened=true;
}
else if (PositionGetInteger (POSITION_TYPE) ==POSITION_TYPE_SELL)
{
SellOpened=true;
}
}
}

//-----
-----
```

```
MqlTradeRequest mrequest;
MqlTradeCheckResult check_result;
MqlTradeResult mresult;

MqlTick latest_price;
if (!SymbolInfoTick (_Symbol, latest_price))
{
    Alert («Ошибка получения последних котировок – ошибка:», GetLastError (),»!!»);
    return;
}
if (Buy==true&&BuyOpened==false) {
// -----
if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest.volume = Lot;
    mrequest.price = NormalizeDouble(latest_price.ask,_Digits);
    mrequest.sl = NormalizeDouble(latest_price.bid – StopLoss,_Digits);
    mrequest.tp = NormalizeDouble(latest_price.ask + Profit,_Digits);
    mrequest.deviation=10;
    mrequest.type = ORDER_TYPE_BUY;
    mrequest.type_filling = ORDER_FILLING_FOK;

    ZeroMemory (check_result);
    ZeroMemory (mresult);
    if (!OrderCheck (mrequest, check_result))
    {
        if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
        if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
        if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
        if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
        return;
    } else {
        if (OrderSend (mrequest, mresult)) {
            if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
            {
                priceBuy=mresult.price;
                slBuy= mrequest.sl;
            }
            else
            {
                if(mresult.retcode==10004) //Реквота
                {
                    Print («Requote bid ",mresult.bid);
```

```
Print («Requote ask ",mresult.ask);
} else {
Print («Retcode ",mresult.retcode);
}
}
} else {
Print («Retcode ",mresult.retcode);
}
}
}
// -----
-----
if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_DEAL;
mrequest.symbol = _Symbol;
mrequest.volume = Lot;
mrequest.type = ORDER_TYPE_BUY;
mrequest.type_filling = ORDER_FILLING_FOK;
ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
return;
} else {
if (OrderSend (mrequest, mresult)) {
if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
{
priceBuy=mresult.price;

// -----
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_SLTP;
mrequest.symbol = _Symbol;
mrequest.sl = NormalizeDouble(mresult.price - StopLoss,_Digits);
mrequest.tp = NormalizeDouble(mresult.price + Profit,_Digits);
ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
return;
} else {
```

```
        if (OrderSend (mrequest, mresult)) {
            if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
            {
                slBuy= mrequest.sl;
            }
            else
            {
                Print («Retcode ",mresult.retcod);
            }
            } else {
                Print («Retcode ",mresult.retcod);
            }
            }
            // -----
        }
        else
        {
            Print («Retcode ",mresult.retcod);
        }
        } else {
            Print («Retcode ",mresult.retcod);
        }
        }
        // -----
    }
    // -----
}

if (SellStop==true&&SellOpened==true) {
    // -----
}

if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest. volume = Lot;
    mrequest.price = NormalizeDouble(latest_price.ask,_Digits);
    mrequest.sl = NormalizeDouble (0.0,_Digits);
    mrequest.tp = NormalizeDouble (0.0,_Digits);
    mrequest.deviation=10;
    mrequest. type = ORDER_TYPE_BUY;
    mrequest. type_filling = ORDER_FILLING_FOK;

    ZeroMemory (check_result);
    ZeroMemory (mresult);
    if (!OrderCheck (mrequest, check_result))
```

```
{
    if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
    if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
    if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
    if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
    return;
} else {
    if (OrderSend (mrequest, mresult)) {
        if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
        {
            Print («Price», mresult.price);
        }
        else
        {
            if(mresult.retcode==10004) //Реквота
            {
                Print («Requote bid ",mresult.bid);
                Print («Requote ask ",mresult.ask);
            } else {
                Print («Retcode ",mresult.retcode);
            }
        }
        } else {
            Print («Retcode ",mresult.retcode);
        }
    }
}
// -----
-----
if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE))==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest.volume = Lot;
    mrequest.type = ORDER_TYPE_BUY;
    mrequest.type_filling = ORDER_FILLING_FOK;

    ZeroMemory (check_result);
    ZeroMemory (mresult);
    if (!OrderCheck (mrequest, check_result))
    {
        if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
        if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
        return;
    } else {
```

```
        if (OrderSend (mrequest, mresult)) {
            if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
            {
                Print («Price», mresult.price);
            }
            else
            {
                Print («Retcode ",mresult.retcod);
            }
        } else {
            Print («Retcode ",mresult.retcod);
        }
    }
}
// -----
}
// -----
}
// -----
if (Sell==true&&SellOpened==false) {
// -----
}
// -----
if ( (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest. volume = Lot;
    mrequest.price = NormalizeDouble(latest_price.bid,_Digits);
    mrequest.sl = NormalizeDouble(latest_price.ask + StopLoss,_Digits);
    mrequest.tp = NormalizeDouble(latest_price.bid – Profit,_Digits);
    mrequest.deviation=10;
    mrequest. type = ORDER_TYPE_SELL;
    mrequest. type_filling = ORDER_FILLING_FOK;

    ZeroMemory (check_result);
    ZeroMemory (mresult);
    if (!OrderCheck (mrequest, check_result))
    {
        if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
        if(check_result.retcod==10015) Alert («Неправильная цена в запросе»);
        if(check_result.retcod==10016) Alert («Неправильные стопы в запросе»);
        if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
        return;
    } else {
        if (OrderSend (mrequest, mresult)) {
```

```
        if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
        {
            priceSell=mresult.price;
            slSell= mrequest.sl;
        }
        else
        {
            if(mresult.retcod==10004) //Реквота
            {
                Print («Requote bid ",mresult.bid);
                Print («Requote ask ",mresult.ask);
            } else {
                Print («Retcode ",mresult.retcod);
            }
        }
        } else {
            Print («Retcode ",mresult.retcod);
        }
    }
}
// -----

if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
    ZeroMemory (mrequest);
    mrequest.action = TRADE_ACTION_DEAL;
    mrequest.symbol = _Symbol;
    mrequest. volume = Lot;
    mrequest. type = ORDER_TYPE_SELL;
    mrequest. type_filling = ORDER_FILLING_FOK;

    ZeroMemory (check_result);
    ZeroMemory (mresult);
    if (!OrderCheck (mrequest, check_result))
    {
        if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
        if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
        return;
    } else {
        if (OrderSend (mrequest, mresult)) {
            if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
            {
                priceSell=mresult.price;

                // -----
                ZeroMemory (mrequest);
```

```
mrequest.action = TRADE_ACTION_SLTP;
mrequest.symbol = _Symbol;
mrequest.tp = NormalizeDouble(mresult.price - Profit,_Digits);
mrequest.sl = NormalizeDouble(mresult.price + StopLoss,_Digits);
ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
return;
} else {
if (OrderSend (mrequest, mresult)) {
if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
{
slSell= mrequest.sl;
}
else
{
Print («Retcode ",mresult.retcode);
}
} else {
Print («Retcode ",mresult.retcode);
}
}
// -----
}
else
{
Print («Retcode ",mresult.retcode);
}
} else {
Print («Retcode ",mresult.retcode);
}
}
}
// -----
-----

}
// -----
-----

if (BuyStop==true&&BuyOpened==true) {
// -----
-----

if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_INSTANT) {
ZeroMemory (mrequest);
```

```
mrequest.action = TRADE_ACTION_DEAL;
mrequest.symbol = _Symbol;
mrequest.volume = Lot;
mrequest.price = NormalizeDouble(latest_price.bid,_Digits);
mrequest.sl = NormalizeDouble (0.0,_Digits);
mrequest.tp = NormalizeDouble (0.0,_Digits);
mrequest.deviation=10;
mrequest.type = ORDER_TYPE_SELL;
mrequest.type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
if(check_result.retcode==10014) Alert («Неправильный объем в запросе»);
if(check_result.retcode==10015) Alert («Неправильная цена в запросе»);
if(check_result.retcode==10016) Alert («Неправильные стопы в запросе»);
if(check_result.retcode==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
return;
} else {
if (OrderSend (mrequest, mresult)) {
if(mresult.retcode==10009 || mresult.retcode==10008) //запрос выполнен или ордер
успешно помещен
{
Print («Price», mresult.price);
}
else
{
if(mresult.retcode==10004) //Пеквота
{
Print («Requote bid ",mresult.bid);
Print («Requote ask ",mresult.ask);
} else {
Print («Retcode ",mresult.retcode);
}
}
} else {
Print («Retcode ",mresult.retcode);
}
}
}
// -----

if (((ENUM_SYMBOL_TRADE_EXECUTION) SymbolInfoInteger (Symbol
(),SYMBOL_TRADE_EXEMODE)) ==SYMBOL_TRADE_EXECUTION_EXCHANGE) {
ZeroMemory (mrequest);
mrequest.action = TRADE_ACTION_DEAL;
mrequest.symbol = _Symbol;
```

```
mrequest. volume = Lot;
mrequest. type = ORDER_TYPE_SELL;
mrequest. type_filling = ORDER_FILLING_FOK;

ZeroMemory (check_result);
ZeroMemory (mresult);
if (!OrderCheck (mrequest, check_result))
{
    if(check_result.retcod==10014) Alert («Неправильный объем в запросе»);
    if(check_result.retcod==10019) Alert («Нет достаточных денежных средств для выпол-
нения запроса»);
    return;
} else {
    if (OrderSend (mrequest, mresult)) {
        if(mresult.retcod==10009 || mresult.retcod==10008) //запрос выполнен или ордер
успешно помещен
        {
            Print («Price», mresult.price);
        }
        else
        {
            Print («Retcode ",mresult.retcod);
        }
        } else {
            Print («Retcode ",mresult.retcod);
        }
        }
        }
        // -----
}
// -----
}

В функции OnInit советника вызовем функцию Trailing класса Trade.
#include "CheckTrade.mqh»
#include "Trade.mqh»
#include "Sidus.mqh»

input double Lot=1;
input double spreadLevel=5.0;
input double StopLoss=0.01;
input double Profit=0.1;
input int numberBarOpenPosition=4;
input int numberBarStopPosition=5;

input double TrailingStop=0.002;

CheckTrade checkTrade;
```

Trade trade (StopLoss, Profit, Lot, TrailingStop);
Sidus sidus (numberBarOpenPosition, numberBarStopPosition);

bool flagStopLoss=false;

//+-----+

|| Expert initialization function |

//+-----+

int OnInit ()

{

return (checkTrade. OnCheckTradeInit (Lot));

}

//+-----+

|| Expert tick function |

//+-----+

void OnTick ()

{

if (!checkTrade. OnCheckTradeTick (Lot, spreadLevel)) {

return;

}

//Ограничить вычисления советника по появлению нового бара на графике

static datetime last_time;

datetime last_bar_time= (datetime) SeriesInfoInteger (Symbol (),Period
(),SERIES_LASTBAR_DATE);

if (last_time!=last_bar_time)

{

last_time=last_bar_time;

} else {

return;

}

/*

//Ограничить вычисления советника по flagStopLoss

static datetime last_time_daily;

datetime last_bar_time_daily= (datetime) SeriesInfoInteger (Symbol
(),PERIOD_D1,SERIES_LASTBAR_DATE);

if (last_time_daily!=last_bar_time_daily)

{

last_time_daily=last_bar_time_daily;

flagStopLoss=false;

}

if (flagStopLoss==true) return;

*/

//Для вычисления сигналов торговой системы требуются исторические данные символа

```
MqlRates mrate [];  
ResetLastError ();  
if (CopyRates (Symbol (),Period (),0,numberBarStopPosition, mrate) <0)  
{  
Print (GetLastError ());  
return;  
}
```

```
ArraySetAsSeries (mrate, true);
```

```
// -----  
-----
```

```
bool TradeSignalBuy=false;  
bool TradeSignalSell=false;
```

```
TradeSignalBuy=sidus. OnTradeSignalBuy ();  
TradeSignalSell=sidus. OnTradeSignalSell ();
```

```
bool TradeSignalBuyStop=false;  
bool TradeSignalSellStop=false;
```

```
//TradeSignalBuyStop=sidus. OnTradeSignalBuyStop ();  
//TradeSignalSellStop=sidus. OnTradeSignalSellStop ();
```

```
trade. Order (TradeSignalBuy, TradeSignalBuyStop, TradeSignalSell, TradeSignalSellStop);  
trade.Trailing ();
```

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

```
//| Trade function |
```

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

```
void OnTrade ()
```

```
{  
static int _deals;  
ulong _ticket=0;
```

```
if (HistorySelect (0,TimeCurrent ()))  
{  
int i=HistoryDealsTotal () -1;
```

```
if (_deals!=i) {  
_deals=i;  
} else {return;}  

```

```
if ((_ticket=HistoryDealGetTicket (i))> 0)
```

```
{  
string _comment=HistoryDealGetString (_ticket, DEAL_COMMENT);  
if (StringFind (_comment, «sl», 0)!=-1) {  
flagStopLoss=true;  
}  
  
}  
}  
}
```

```
// -----  
// Expert deinitialization function |  
//+-----+  
void OnDeinit (const int reason)  
{  
  
}
```

Матожидание выигрыша теперь $230.89 - 179.78 = 51$.

Хотя чистая прибыль при трейдинге увеличилась, статистическая прибыльность эксперта ухудшилась.

Другой показатель вкладки Бэктест тестера стратегий, это Коэффициент Шарпа, характеризующий эффективность и стабильность эксперта. Чем выше значение показателя, тем больше доходность на единицу риска. Значение коэффициента Шарпа для устойчивых экспертов более 0.25.

В нашем случае коэффициент Шарпа равен 0.09.

Фактор роста эксперта или GHPR (среднее геометрическое сделки) будет равен (Конечный депозит/Начальный депозит) в степени 1/ (Всего трейдов).

В данном случае GHPR (фактор роста) равен 1,0036 – больше единицы, что означает возможность торговли с использованием реинвестирования.

Другие показатели эксперта, которые можно увидеть во вкладке Бэктест тестера стратегий, это:

Прибыльность (Profit Factor) – Общая прибыль/общий убыток. Рекомендуемое значение не меньше 2.

Фактор восстановления – Чистая прибыль / Максимальная просадка по средствам. Чем больше показатель, тем менее рискованным является советник. Многие эксперты считают, что у эффективной торговой системы фактор восстановления должен быть не менее 3.

АНPR – среднеарифметическое сделки. Положительное значение говорит о том, что торговая система прибыльна.

Уровень маржи – минимальный уровень маржи в процентах, который был зафиксирован за период тестирования.

LR Correlation – позволяет оценить отклонения точек графика баланса счета от линейной регрессии. Чем LR Correlation ближе к нулю, тем более случайный характер имеет торговля.

LR Standard Error – отклонение графика баланса счета от линейной регрессии в денежном выражении.

Средний прибыльный трейд / Средний убыточный трейд – желательно, чтобы отношение этих двух показателей было больше единицы.

Максимальное количество непрерывных проигрышей (убыток) – желательно, чтобы этот показатель был как можно меньше.

Исходя из глубины максимальной посадки по средствам, можно вычислить минимальный депозит, необходимый, чтобы начать торговать с данным экспертом.

Мин. депозит = Максимальная просадка по средствам * 2

Correlation (Profits, MFE) – связь между результатами позиций и MFE (Maximum Favorable Excursion – максимальный размер потенциальной прибыли, наблюдаемый за время удержания позиции). MFE показывает максимальное движение цены в благоприятном направлении. Чем ближе показатель Correlation (Profits, MFE) к единице, тем лучше эксперт реализует потенциальную прибыль.

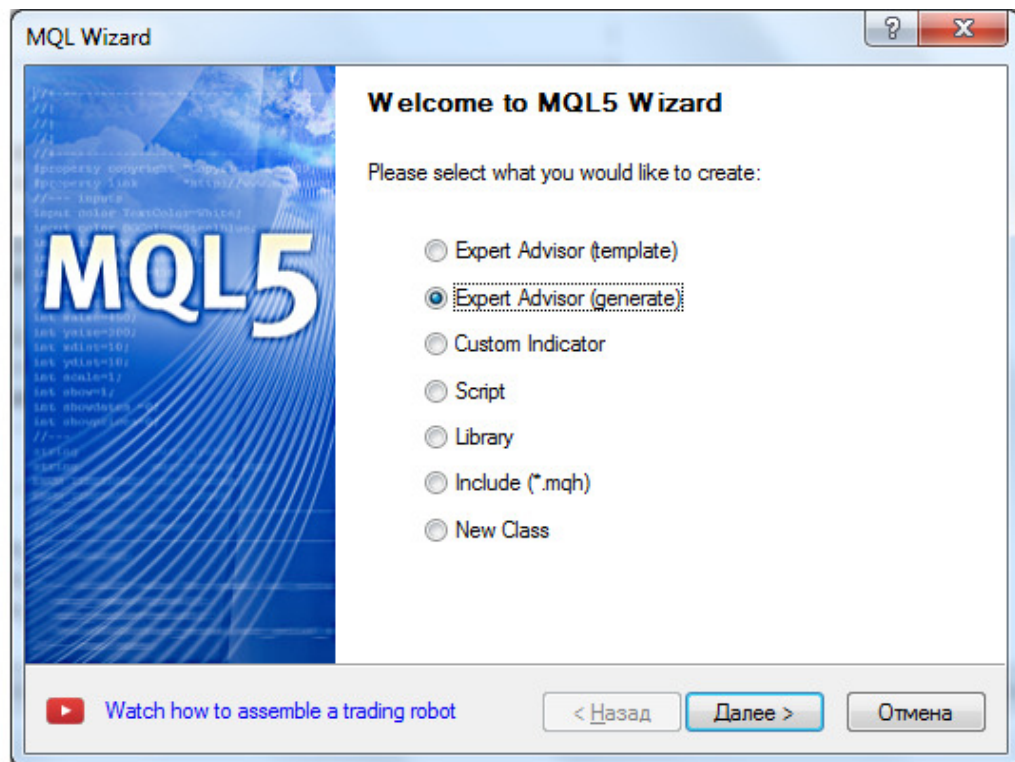
Correlation (Profits, MAE) – связь между результатами позиций и MAE (Maximum Adverse Excursion – максимальный потенциальный убыток, наблюдаемый за время удержания позиции). MAE показывает максимально неблагоприятное движение цены. Чем ближе показатель Correlation (Profits, MAE) к единице, тем лучше эксперт использует защитный Stop Loss.

Correlation (MFE, MAE) – связь между MFE и MAE. Чем ближе показатель Correlation (MFE, MAE) к единице, тем лучше эксперт реализует максимальную прибыль и максимально защищает позицию на всем протяжении ее жизни.

Среднее время удержания позиции – показатель рассчитывается как общее время удержания, деленное на количество сделок. Время удержания позиции увеличивает риск, потому что просадка, очевидно, может быть большей при позициях, удерживаемых в течение большего периода времени.

Создание эксперта с помощью мастера MQL5 Wizard

Мастер MQL5 Wizard, который открывается с помощью кнопки New панели инструментов редактора MetaEditor, позволяет сгенерировать код эксперта на основе готовых модулей – сигналов, управления капиталом и трейлинг-стопа.



MQL Wizard



General properties of the Expert Advisor
Please specify general properties of the Expert Advisor.


Name:

Author:

Link:

Parameters:


Name	Type	Value
 Symbol	string	current
 TimeFrame	Timeframe	current

 [Watch how to assemble a trading robot](#)

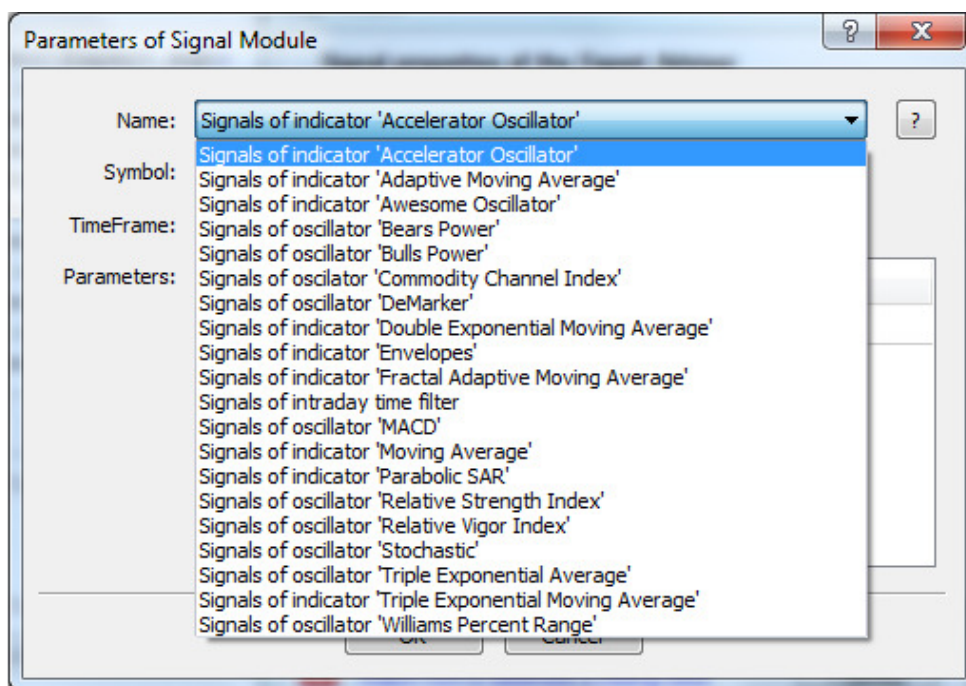
MQL Wizard

Signal properties of the Expert Advisor
Please specify signal properties of the Expert Advisor.

Selected signals	Weight	Period	Symbol
------------------	--------	--------	--------

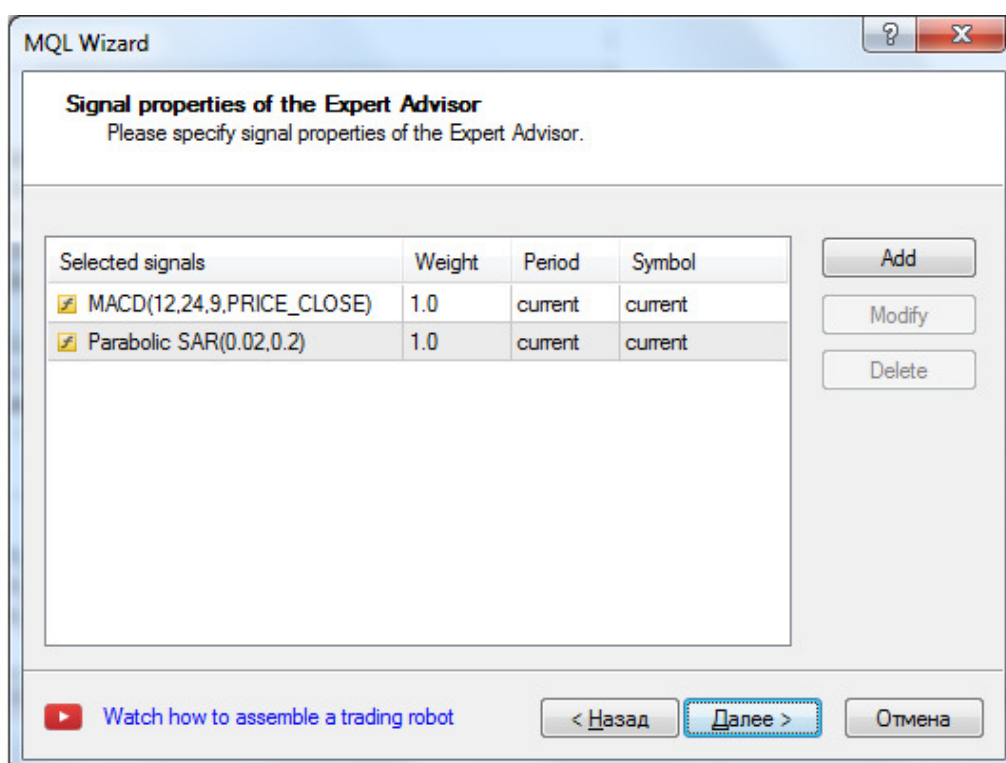
 [Watch how to assemble a trading robot](#)

Модуль сигнала добавляется с помощью кнопки Add.



Файлы модулей сигналов – это включаемые файлы Include (*.mqh), расположенные в папке MQL5\Include\Expert\Signal.

В качестве примера выберем сигнал MACD и сигнал PSAR.



Если посмотреть файлы SignalMACD.mqh и SignalSAR.mqh папки MQL5\Include\Expert\Signal, сигнал MACD имеет 5 моделей прогноза цены:

Модель 0 – «осциллятор имеет необходимое направление» – значимость 10.

Модель 1 – «разворот осциллятора в нужном направлении» – значимость 30.

Модель 2 – «пересечение основной и сигнальной линии» – значимость 80.

Модель 3 – «пересечение главной линией нулевого уровня» – значимость 50.

Модель 4 – «дивергенция осциллятора и цены» – значимость 60.

Модель 5 – «двойная дивергенция осциллятора и цены» – значимость 100.

Сигнал SAR имеет 2 модели прогноза цены:

Модель 0 – «параболик находится на нужной стороне цены» – значимость 40.

Модель 1 – «параболик переключается на другую сторону цены» – значимость 90.

Если модель дает сигнал на падение цены – значимость отрицательная, если на рост цены – значимость положительная.

Итоговый прогноз двух модулей будет рассчитываться по формуле:

Итоговый Прогноз = (Прогноз MACD + Прогноз SAR) / 2

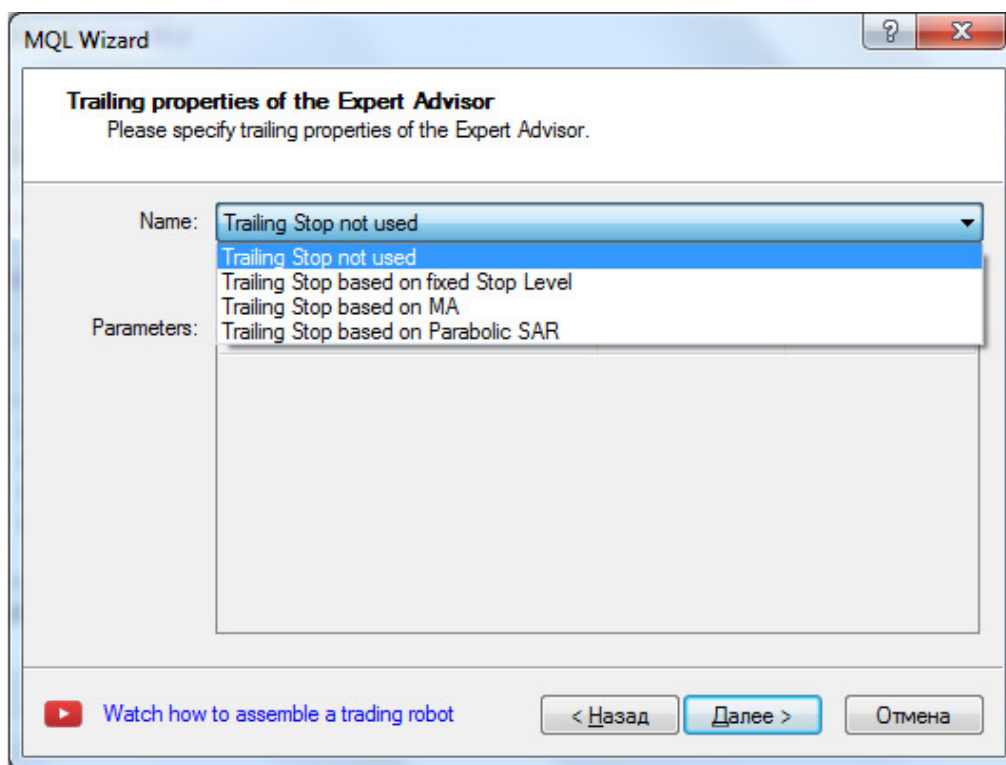
Где Прогноз MACD = Вес сигнала MACD * значимость Модели MACD,

Прогноз SAR = Вес сигнала SAR * значимость Модели SAR

В нашем случае мы установили веса сигналов равными единице.

Если итоговый прогноз превысит пороговое значение, эксперт совершит сделку на покупку или продажу.

После определения сигналов эксперта определяется алгоритм сопровождения открытой позиции.



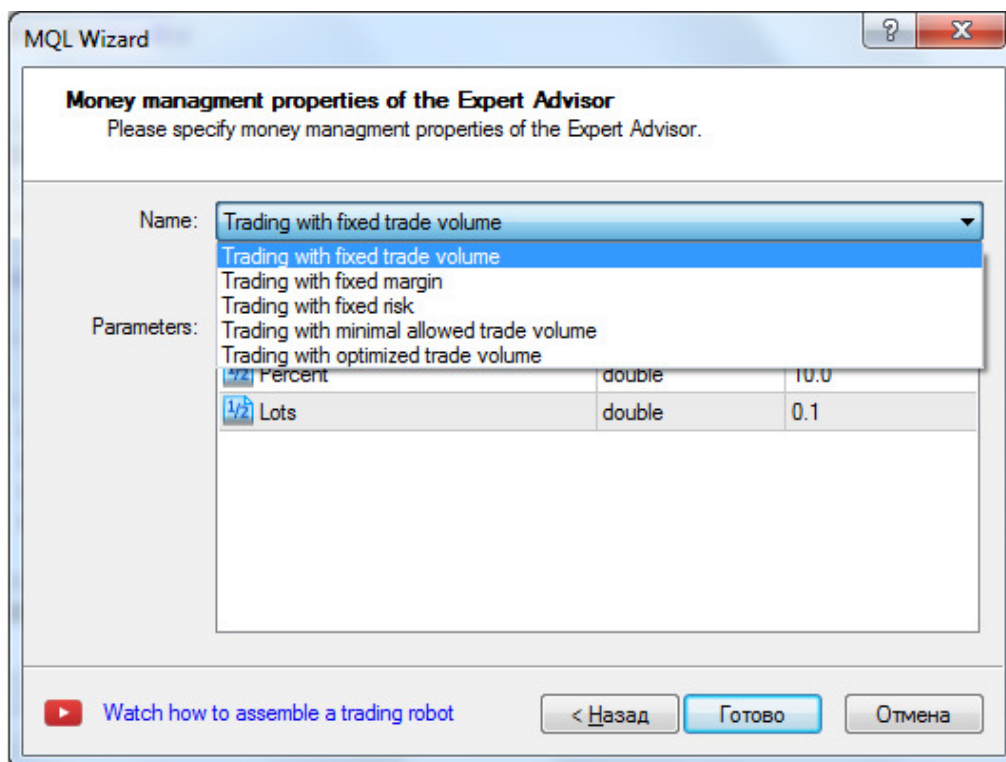
– Сопровождение открытой позиции на фиксированном «расстоянии» (в пунктах) – уровни Stop Loss и Take Profit открытой позиции перемещаются на фиксированное расстояние по движению цены в направлении открытой позиции. Когда цена перемещается в направлении открытой позиции на расстояние, которое превышает количество пунктов, соответствующих уровню Trailing Stop Level, эксперт изменяет значения уровней Stop Loss и Take Profit (если Trailing Profit Level > 0).

– Сопровождение открытой позиции по значениям скользящей средней на предыдущем баре.

– Сопровождение открытой позиции по значениям индикатора Parabolic SAR на предыдущем баре.

Файлы реализации алгоритма сопровождения открытой позиции находятся в папке MQL5\Include\Expert\Trailing.

После определения алгоритма сопровождения открытой позиции устанавливается алгоритм управления капиталом и рисками.



– Торговля с фиксированным лотом.

– Торговля с фиксированным уровнем маржи. Значение лота вычисляется функцией MaxLotCheck, которая возвращает максимально возможный объем торговой операции на основе доли свободной маржи (здесь по умолчанию 10%).

– Торговля с фиксированным уровнем риска. Значение лота вычисляется как отношение доли баланса, выделенной для риска, к StopLoss.

– Торговля минимальным лотом.

– Торговля объемом, определяемым результатами предыдущих сделок. Здесь сначала значение лота вычисляется функцией MaxLotCheck, которая возвращает максимально возможный объем торговой операции на основе доли свободной маржи (здесь по умолчанию 10%). Затем в случае получения убытка лот уменьшается на фактор Decrease Factor (по умолчанию 3).

Если образуется убыток, равный указанному проценту от текущего капитала, производится принудительное закрытие убыточной позиции.

Файлы реализации алгоритма управления капиталом и рисками находятся в папке MQL5\Include\Expert\Money.

После выбора алгоритма управления капиталом и рисками генерируется код эксперта.

Код эксперта основан на использовании экземпляра класса CExpert, файл которого находится в папке MQL5\Include\Expert.

Пороговые значения Signal_ThresholdOpen и Signal_ThresholdClose итогового прогноза сигналов по умолчанию равны 10.

Тестирование этого эксперта с сигналами MACD и PSAR на часовом графике EUR/USD с разными алгоритмами трейлинга и манименеджмента дает отрицательное матожидание выигрыша. Ничего не дает и оптимизация таких параметров, как Trailing_FixedPips_StopLevel, Signal_MACD_Weight и Signal_SAR_Weight. Эксперт остается убыточным.

Попробуем создать эксперта и включить в него все стандартные сигналы мастера MQL5 Wizard.

При тестировании на часовом графике EUR/USD такой эксперт с равными весами сигналов также покажет отрицательное матожидание выигрыша.

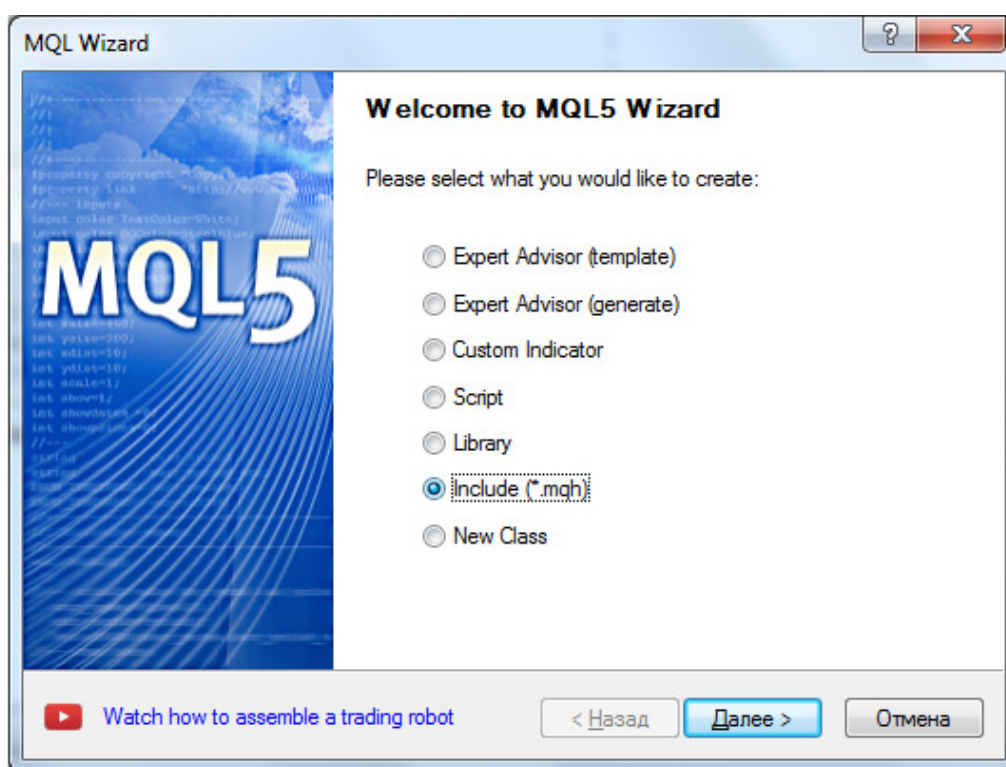
Однако при оптимизации весов сигналов, эксперт выйдет в плюс, и будут видны комбинации сигналов со значимыми весами, дающие наилучший результат. Также можно провести оптимизацию пороговых значений прогноза вместе с оптимизацией весов сигналов.

На основании полученных комбинаций сигналов, выводящих советник в прибыль, уже можно строить торговые системы.

Модульная структура эксперта, генерируемого мастером MQL5 Wizard, позволяет включить в советник свой собственный модуль сигналов торговой системы.

В качестве примера рассмотрим создание модуля сигналов на основе торговой системы Сидуса.

В редакторе MetaEditor нажмем кнопку New и в мастере MQL5 Wizard создадим включаемый файл.



Который поместим в папку MQL5\Include\Expert\Signal.

Включаемый файл должен содержать класс, расширяющий класс CExpertSignal. Поэтому в код необходимо включить файл ExpertSignal.mqh класса CExpertSignal.

```
#include <Expert\ExpertSignal.mqh>
```

Далее должна присутствовать информация о модуле сигналов, предназначенная для мастера MQL5 Wizard, которая используется для распознавания модуля сигналов мастером MQL5 Wizard при создании эксперта в окне добавления сигналов, а также при генерации

самого кода эксперта. Без этой информации мастер MQL5 Wizard просто не добавит сигнал в свой интерфейс, хотя файл сигнала и будет расположен в папке MQL5\Include\Expert\Signal. После создания модуля сигналов редактор MetaEditor необходимо перезагрузить, чтобы сигнал добавился в MQL5 Wizard.

```
// wizard description start
//+-----+
//| Description of the class |
//| Title=Signals of the system «Sidus» |
//| Type=SignalAdvanced |
//| Name=Sidus |
//| ShortName=MA |
//| Class=CSignalSidus |
//| Page= |
//| Parameter=NumberOpenPosition, int,5,Bars number checked to cross |
//| Parameter=Pattern_0,int,80,Model 0 |
//| Parameter=Pattern_1,int,20,Model 1 |
//| Parameter=Pattern_2,int,80,Model 2 |
//+-----+
// wizard description end
//+-----+
//| Class CSignalSidus. |
//| Purpose: Class of generator of trade signals based on |
//| the «Sidus» system. |
//| Is derived from the CExpertSignal class. |
//+-----+
```

Здесь строка Title отображается в списке сигналов окна MQL5 Wizard, строка Page указывает на раздел справки и должна быть пустой, строки Parameter описывают методы установки параметров сигнала и используются при генерации кода советника.

Далее идет объявление параметров и методов класса модуля сигналов.

```
class CSignalSidus: public CExpertSignal
{
protected:
    CiMA m_ma18; // object-indicator
    CiMA m_ma28; // object-indicator
    CiMA m_ma5; // object-indicator
    CiMA m_ma8; // object-indicator
    // - - adjusted parameters
    int m_numberOpenPosition;
    // - - «weights» of market models (0—100)
    int m_pattern_0; // model 0 «5 WMA and 8 WMA cross the 18 EMA and the 28 EMA upward
or top down» 80
    int m_pattern_1; // model 1 «5 WMA crosses the 8 WMA upward or top down» 10
    int m_pattern_2; // model 2 «18 EMA crosses the 28 EMA upward or top down» 80

public:
    CSignalSidus (void);
    ~CSignalSidus (void);
    // - - methods of setting adjustable parameters
    void NumberOpenPosition (int value) {m_numberOpenPosition=value;}
```

```
// -- methods of adjusting «weights» of market models
void Pattern_0 (int value) {m_pattern_0=value;}
void Pattern_1 (int value) {m_pattern_1=value;}
void Pattern_2 (int value) {m_pattern_2=value;}
// -- method of verification of settings
virtual bool ValidationSettings (void);
// -- method of creating the indicator and timeseries
virtual bool InitIndicators (CIndicators *indicators);
// -- methods of checking if the market models are formed
virtual int LongCondition (void);
virtual int ShortCondition (void);
```

protected:

```
// -- method of initialization of the indicator
bool InitMA (CIndicators *indicators);
```

```
};
```

Здесь объекты `m_ma*` представляют скользящие средние, используемые системой Сидуса, параметр `m_numberOpenPosition` представляет количество баров, на которых будет проверяться пересечение скользящих средних.

Для генерации сигналов определим три модели прогноза цены – пересечение скользящих средних 5WMA и 8 WMA скользящих средних 18 EMA и 28 EMA, пересечение скользящей средней 5WMA скользящую среднюю 8 WMA, пересечение скользящей средней 18 EMA скользящую среднюю 28 EMA.

Метод `ValidationSettings` проверяет на корректность параметры сигнала, метод `InitIndicators` инициализирует объекты `m_ma*`.

Эксперт получает сигналы модуля с помощью методов `CheckOpenLong`, `CheckOpenShort`, `CheckCloseLong`, `CheckCloseShort`, `CheckReverseLong`, `CheckReverseShort` класса `CExpertSignal`.

Однако эти методы, в свою очередь, формируют свои возвращаемые значения на основе значений, которые возвращаются методами `LongCondition` и `ShortCondition`.

Методы `LongCondition` и `ShortCondition` как раз и оперируют моделями прогноза цены. Потому в модуле сигналов достаточно определить эти два метода.

Определим конструктор класса и его методы.

```
CSignalSidus::CSignalSidus (void): m_numberOpenPosition (5),
m_pattern_0 (80),
m_pattern_1 (10),
m_pattern_2 (80)
{
// -- initialization of protected data
m_used_series=USE_SERIES_OPEN+USE_SERIES_HIGH+USE_SERIES_LOW
+USE_SERIES_CLOSE;
}
//+-----+
//| Destructor |
//+-----+
CSignalSidus::~CSignalSidus (void)
{
}
```

```

//+-----+
// Validation settings protected data. |
//+-----+
bool CSignalSidus::ValidationSettings (void)
{
// -- validation settings of additional filters
if (!CExpertSignal::ValidationSettings ())
return (false);

// -- ok
return (true);
}
//+-----+
// Create indicators. |
//+-----+
bool CSignalSidus::InitIndicators (CIndicators *indicators)
{
// -- check pointer
if (indicators==NULL)
return (false);
// -- initialization of indicators and timeseries of additional filters
if (!CExpertSignal::InitIndicators (indicators))
return (false);
// -- create and initialize MA indicator
if (!InitMA (indicators))
return (false);
// -- ok
return (true);
}
//+-----+
// Initialize MA indicators. |
//+-----+
bool CSignalSidus::InitMA (CIndicators *indicators)
{
// -- check pointer
if (indicators==NULL)
return (false);
// -- add object to collection
if (!indicators.Add (GetPointer (m_ma18)))
{
printf (__FUNCTION__+": error adding object»);
return (false);
}
// -- initialize object
if (!m_ma18.Create(m_symbol.Name
PRICE_WEIGHTED))
{
printf (__FUNCTION__+": error initializing object»);
return (false);
},m_period,18,0,MODE_EMA,

```

```

    }
    // -- add object to collection
    if(!indicators.Add (GetPointer (m_ma28)))
    {
        printf (__FUNCTION__+": error adding object»);
        return (false);
    }
    // -- initialize object
    if(!m_ma28.Create(m_symbol.Name                                (),m_period,28,0,MODE_EMA,
PRICE_WEIGHTED))
    {
        printf (__FUNCTION__+": error initializing object»);
        return (false);
    }
    // -- add object to collection
    if(!indicators.Add (GetPointer (m_ma5)))
    {
        printf (__FUNCTION__+": error adding object»);
        return (false);
    }
    // -- initialize object
    if(!m_ma5.Create(m_symbol.Name                                (),m_period,5,0,MODE_LWMA,
PRICE_WEIGHTED))
    {
        printf (__FUNCTION__+": error initializing object»);
        return (false);
    }
    // -- add object to collection
    if(!indicators.Add (GetPointer (m_ma8)))
    {
        printf (__FUNCTION__+": error adding object»);
        return (false);
    }
    // -- initialize object
    if(!m_ma8.Create(m_symbol.Name                                (),m_period,8,0,MODE_LWMA,
PRICE_WEIGHTED))
    {
        printf (__FUNCTION__+": error initializing object»);
        return (false);
    }
    // -- ok
    return (true);
}
//+-----+
//| «Voting» that price will grow. |
//+-----+
int CSignalSidus::LongCondition (void)
{
    int result=0;

```

```
int idx=StartIndex ();

if(m_ma5.Main(idx)>m_ma8.Main(idx)&&m_ma8.Main(idx)>m_ma18.Main(idx)&&m_ma8.Main
bool flagCross1=false;
bool flagCross2=false;
for (int i= (idx+1);i <m_numberOpenPosition; i++) {
if(m_ma5.Main(i)<m_ma18.Main(i)&&m_ma5.Main(i)<m_ma28.Main (i)) {
flagCross1=true;
}
if(m_ma8.Main(i)<m_ma18.Main(i)&&m_ma8.Main(i)<m_ma28.Main (i)) {
flagCross2=true;
}
}
if (flagCross1==true&&flagCross2==true) {
result=m_pattern_0;

}
}

if(m_ma5.Main(idx)>m_ma8.Main (idx)) {
bool flagCross=false;
for (int i= (idx+1);i <m_numberOpenPosition; i++) {
if(m_ma5.Main(i)<m_ma8.Main (i)) {
flagCross=true;
}
}
if (flagCross==true) {
result=m_pattern_1;
}
}

if(m_ma18.Main(idx)>m_ma28.Main (idx)) {
bool flagCross=false;
for (int i= (idx+1);i <m_numberOpenPosition; i++) {
if(m_ma18.Main(i)<m_ma28.Main (i)) {
flagCross=true;
}
}
if (flagCross==true) {
result=m_pattern_2;
}
}

return result;
}
//+-----+
//| «Voting» that price will fall. |
//+-----+
int CSignalSidus::ShortCondition (void)
```



```
//| Include |
//+-----+
#include <Expert\Expert.mqh>
// - - available signals
#include <Expert\Signal\SignalSidus.mqh>
// - - available trailing
#include <Expert\Trailing\TrailingFixedPips.mqh>
// - - available money management
#include <Expert\Money\MoneyFixedLot.mqh>
//+-----+
//| Inputs |
//+-----+
// - - inputs for expert
input string Expert_Title =«EMTSSGeneratedSidus»; // Document name
ulong Expert_MagicNumber =7506; //
bool Expert_EveryTick =false; //
// - - inputs for main signal
input int Signal_ThresholdOpen =20; // Signal threshold value to open [0...100]
input int Signal_ThresholdClose =20; // Signal threshold value to close [0...100]
input double Signal_PriceLevel =0.0; // Price level to execute a deal
input double Signal_StopLevel =50.0; // Stop Loss level (in points)
input double Signal_TakeLevel =50.0; // Take Profit level (in points)
input int Signal_Expiration =4; // Expiration of pending orders (in bars)
input int Signal_MA_NumberOpenPosition =5; // Sidus (5,80,20,80) Bars number checked
to cross
input int Signal_MA_Pattern_0 =80; // Sidus (5,80,20,80) Model 0
input int Signal_MA_Pattern_1 =20; // Sidus (5,80,20,80) Model 1
input int Signal_MA_Pattern_2 =80; // Sidus (5,80,20,80) Model 2
input double Signal_MA_Weight =1.0; // Sidus (5,80,20,80) Weight [0...1.0]
// - - inputs for trailing
input int Trailing_FixedPips_StopLevel =30; // Stop Loss trailing level (in points)
input int Trailing_FixedPips_ProfitLevel =50; // Take Profit trailing level (in points)
// - - inputs for money
input double Money_FixLot_Percent =10.0; // Percent
input double Money_FixLot_Lots =0.1; // Fixed volume
//+-----+
//| Global expert object |
//+-----+
CExpert ExtExpert;
//+-----+
//| Initialization function of the expert |
//+-----+
int OnInit ()
{
// - - Initializing expert
if(!ExtExpert.Init (Symbol (),Period (),Expert_EveryTick, Expert_MagicNumber))
{
// - - failed
printf (__FUNCTION__+"": error initializing expert»);
}
```

```
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- Creating signal
CExpertSignal *signal=new CExpertSignal;
if (signal==NULL)
{
// -- failed
printf (__FUNCTION__+": error creating signal»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// --
ExtExpert.InitSignal (signal);
signal.ThresholdOpen (Signal_ThresholdOpen);
signal.ThresholdClose (Signal_ThresholdClose);
signal.PriceLevel (Signal_PriceLevel);
signal.StopLevel (Signal_StopLevel);
signal.TakeLevel (Signal_TakeLevel);
signal.Expiration (Signal_Expiration);
// -- Creating filter CSignalSidus
CSignalSidus *filter0=new CSignalSidus;
if (filter0==NULL)
{
// -- failed
printf (__FUNCTION__+": error creating filter0»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
signal.AddFilter (filter0);
// -- Set filter parameters
filter0.NumberOpenPosition (Signal_MA_NumberOpenPosition);
filter0.Pattern_0 (Signal_MA_Pattern_0);
filter0.Pattern_1 (Signal_MA_Pattern_1);
filter0.Pattern_2 (Signal_MA_Pattern_2);
filter0.Weight (Signal_MA_Weight);
// -- Creation of trailing object
CTrailingFixedPips *trailing=new CTrailingFixedPips;
if (trailing==NULL)
{
// -- failed
printf (__FUNCTION__+": error creating trailing»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- Add trailing to expert (will be deleted automatically)
if (!ExtExpert.InitTrailing (trailing))
{
// -- failed
```

```
printf (__FUNCTION__+": error initializing trailing»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- Set trailing parameters
trailing.StopLevel (Trailing_FixedPips_StopLevel);
trailing.ProfitLevel (Trailing_FixedPips_ProfitLevel);
// -- Creation of money object
CMoneyFixedLot *money=new CMoneyFixedLot;
if (money==NULL)
{
// -- failed
printf (__FUNCTION__+": error creating money»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- Add money to expert (will be deleted automatically))
if(!ExtExpert.InitMoney (money))
{
// -- failed
printf (__FUNCTION__+": error initializing money»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- Set money parameters
money.Percent (Money_FixLot_Percent);
money.Lots (Money_FixLot_Lots);
// -- Check all trading objects parameters
if(!ExtExpert.ValidationSettings ())
{
// -- failed
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- Tuning of all necessary indicators
if(!ExtExpert.InitIndicators ())
{
// -- failed
printf (__FUNCTION__+": error initializing indicators»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- ok
return (INIT_SUCCEEDED);
}
//+-----+
//| Deinitialization function of the expert |
//+-----+
void OnDeinit (const int reason)
```

```

{
ExtExpert.Deinit ();
}
//+-----+
//| «Tick» event handler function |
//+-----+
void OnTick ()
{
ExtExpert. OnTick ();
}
//+-----+
//| «Trade» event handler function |
//+-----+
void OnTrade ()
{
ExtExpert. OnTrade ();
}
//+-----+
//| «Timer» event handler function |
//+-----+
void OnTimer ()
{
ExtExpert. OnTimer ();
}
//+-----+

```

При оптимизации параметров данного эксперта на часовом графике EUR/USD получим, что эксперт лучше всего работает со следующими значениями параметров:

```

input int Signal_ThresholdOpen =20; // Signal threshold value to open [0...100]
input int Signal_ThresholdClose =20; // Signal threshold value to close [0...100]
input int Signal_MA_NumberOpenPosition =3; // Sidus (5,80,20,80) Bars number checked
to cross
input int Signal_MA_Pattern_0 =60; // Sidus (5,80,20,80) Model 0
input int Signal_MA_Pattern_1 =10; // Sidus (5,80,20,80) Model 1
input int Signal_MA_Pattern_2 =100; // Sidus (5,80,20,80) Model 2
input int Trailing_FixedPips_StopLevel =100; // Stop Loss trailing level (in points)

```

Создание индикатора на основе модулей торговых сигналов эксперта

В качестве примера создадим индикатор на основе двух модулей сигналов SignalMA и SignalMACD, файлы которых находятся в каталоге MQL5\Include\Expert\Signal.

В мастере MQL5 Wizard создадим основу индикатора, выбрав вариант Custom Indicator.

Теперь нам нужно указать свойства индикатора, а также определить функции OnInit и OnCalculate.

При создании индикатора на основе модулей торговых сигналов эксперта будем опираться на код сгенерированного с помощью MQL5 Wizard эксперта.

При работе такого эксперта, в функции OnTick, вызывается функция OnTick класса CExpert.

В этой функции сначала вызывается функция Refresh, обновляющая цены символа и значения индикаторов, а уже потом вызывается функция Processing, которая получает торговые сигналы и выставляет ордера.

Так как функция Refresh является защищенной, а нам нужно обновлять данные модулей торговых сигналов в нашей функции OnCalculate, создадим свой класс CExpertInd, расширяющий класс CExpert.

```
#include <Expert\Expert.mqh>
```

```
class CExpertInd: public CExpert
{
public:
virtual bool RefreshInd (void);
};
```

```
//+-----+
//| Refreshing data for processing |
//+-----+
bool CExpertInd::RefreshInd (void)
{
MqlDateTime time;
// - - refresh rates
if(!m_symbol.RefreshRates ())
return (false);
// - - check need processing
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;
// - - refresh indicators
m_indicators.Refresh ();
// - - ok
return (true);
}
```

Посмотрев на классы CSignalMA и CSignalMACD, мы увидим, что функции LongCondition и ShortCondition, дающие рыночные модели, вызываются на текущем баре. Нам же нужно вызывать эти функции на всей истории символа. Кроме того, нам нужна функция BarsCalculated, возвращающая количество рассчитанных значений в модуле сигналов. Поэтому создадим классы CSignalMAInd и CSignalMACDInd, расширяющие классы CSignalMA и CSignalMACD.

```
#include <Expert\Signal\SignalMA.mqh>
class CSignalMAInd: public CSignalMA
{
public:
virtual int BarsCalculatedInd ();
virtual int LongConditionInd (int ind);
virtual int ShortConditionInd (int ind);
};

//+-----+
//| Refresh indicators. |
//+-----+
int CSignalMAInd::BarsCalculatedInd () {
return m_ma.BarsCalculated ();
}

//+-----+
//| «Voting» that price will grow. |
//+-----+
int CSignalMAInd::LongConditionInd (int idx)
{

int result=0;

// - - analyze positional relationship of the close price and the indicator at the first analyzed bar
if (DiffCloseMA (idx) <0.0)
{
// - - the close price is below the indicator
if (IS_PATTERN_USAGE (1) && DiffOpenMA (idx)> 0.0 && DiffMA (idx)> 0.0)
{
// - - the open price is above the indicator (i.e. there was an intersection), but the indicator
is directed upwards
result=m_pattern_1;
// - - consider that this is an unformed «piercing» and suggest to enter the market at the
current price
m_base_price=0.0;
}
}
else
{
// - - the close price is above the indicator (the indicator has no objections to buying)
if (IS_PATTERN_USAGE (0))
result=m_pattern_0;
```

```
// - - if the indicator is directed upwards
if (DiffMA (idx)> 0.0)
{
    if (DiffOpenMA (idx) <0.0)
    {
        // - - if the model 2 is used
        if (IS_PATTERN_USAGE (2))
        {
            // - - the open price is below the indicator (i.e. there was an intersection)
            result=m_pattern_2;
            // - - suggest to enter the market at the «roll back»
            m_base_price=m_symbol.NormalizePrice (MA (idx));
        }
    }
    else
    {
        // - - if the model 3 is used and the open price is above the indicator
        if (IS_PATTERN_USAGE (3) && DiffLowMA (idx) <0.0)
        {
            // - - the low price is below the indicator
            result=m_pattern_3;
            // - - consider that this is a formed «piercing» and suggest to enter the market at the current
price
            m_base_price=0.0;
        }
    }
}
// - - return the result
return (result);
}
//+-----+
//| «Voting» that price will fall. |
//+-----+
int CSignalMAInd::ShortConditionInd (int idx)
{
    int result=0;

    // - - analyze positional relationship of the close price and the indicator at the first analyzed bar
    if (DiffCloseMA (idx)> 0.0)
    {
        // - - the close price is above the indicator
        if (IS_PATTERN_USAGE (1) && DiffOpenMA (idx) <0.0 && DiffMA (idx) <0.0)
        {
            // - - the open price is below the indicator (i.e. there was an intersection), but the indicator
is directed downwards
            result=m_pattern_1;
            // - - consider that this is an unformed «piercing» and suggest to enter the market at the
current price
```

```

        m_base_price=0.0;
    }
}
else
{
    // -- the close price is below the indicator (the indicator has no objections to buying)
    if (IS_PATTERN_USAGE (0))
        result=m_pattern_0;
    // -- the indicator is directed downwards
    if (DiffMA (idx) <0.0)
    {
        if (DiffOpenMA (idx)> 0.0)
        {
            // -- if the model 2 is used
            if (IS_PATTERN_USAGE (2))
            {
                // -- the open price is above the indicator (i.e. there was an intersection)
                result=m_pattern_2;
                // -- suggest to enter the market at the «roll back»
                m_base_price=m_symbol.NormalizePrice (MA (idx));
            }
        }
    }
    else
    {
        // -- if the model 3 is used and the open price is below the indicator
        if (IS_PATTERN_USAGE (3) && DiffHighMA (idx)> 0.0)
        {
            // -- the high price is above the indicator
            result=m_pattern_3;
            // -- consider that this is a formed «piercing» and suggest to enter the market at the current
price
            m_base_price=0.0;
        }
    }
}
// -- return the result
return (result);
}
//+-----+
#include <Expert\Signal\SignalMACD.mqh>

class CSignalMACDInd: public CSignalMACD
{
public:
    virtual int BarsCalculatedInd ();
    virtual int LongConditionInd (int ind);
    virtual int ShortConditionInd (int ind);

```

```

};
//+-----+
// Refresh indicators. |
//+-----+
int CSignalMACDInd::BarsCalculatedInd () {
return m_MACD.BarsCalculated ();
}

//+-----+
// «Voting» that price will grow. |
//+-----+
int CSignalMACDInd::LongConditionInd (int idx)
{
int result=0;

// - - check direction of the main line
if (DiffMain (idx)> 0.0)
{
// - - the main line is directed upwards, and it confirms the possibility of price growth
if (IS_PATTERN_USAGE (0))
result=m_pattern_0; // «confirming» signal number 0
// - - if the model 1 is used, look for a reverse of the main line
if (IS_PATTERN_USAGE (1) && DiffMain (idx+1) <0.0)
result=m_pattern_1; // signal number 1
// - - if the model 2 is used, look for an intersection of the main and signal line
if (IS_PATTERN_USAGE (2) && State (idx)> 0.0 && State (idx+1) <0.0)
result=m_pattern_2; // signal number 2
// - - if the model 3 is used, look for an intersection of the main line and the zero level
if (IS_PATTERN_USAGE (3) && Main (idx)> 0.0 && Main (idx+1) <0.0)
result=m_pattern_3; // signal number 3
// - - if the models 4 or 5 are used and the main line turned upwards below the zero level,
look for divergences
if ((IS_PATTERN_USAGE (4) || IS_PATTERN_USAGE (5)) && Main (idx) <0.0)
{
// - - perform the extended analysis of the oscillator state
ExtState (idx);
// - - if the model 4 is used, look for the «divergence» signal
if (IS_PATTERN_USAGE (4) && CompareMaps (1,1)) // 0000 0001b
result=m_pattern_4; // signal number 4
// - - if the model 5 is used, look for the «double divergence» signal
if (IS_PATTERN_USAGE (5) && CompareMaps (0x11,2)) // 0001 0001b
return (m_pattern_5); // signal number 5
}
}
// - - return the result
return (result);
}
//+-----+
// «Voting» that price will fall. |

```

```
//+-----+
int CSignalMACDInd::ShortConditionInd (int idx)
{
    int result=0;

    // - - check direction of the main line
    if (DiffMain (idx) <0.0)
    {
        // - - main line is directed downwards, confirming a possibility of falling of price
        if (IS_PATTERN_USAGE (0))
            result=m_pattern_0; // «confirming» signal number 0
        // - - if the model 1 is used, look for a reverse of the main line
        if (IS_PATTERN_USAGE (1) && DiffMain (idx+1)> 0.0)
            result=m_pattern_1; // signal number 1
        // - - if the model 2 is used, look for an intersection of the main and signal line
        if (IS_PATTERN_USAGE (2) && State (idx) <0.0 && State (idx+1)> 0.0)
            result=m_pattern_2; // signal number 2
        // - - if the model 3 is used, look for an intersection of the main line and the zero level
        if (IS_PATTERN_USAGE (3) && Main (idx) <0.0 && Main (idx+1)> 0.0)
            result=m_pattern_3; // signal number 3
        // - - if the models 4 or 5 are used and the main line turned downwards above the zero level,
        look for divergences
        if ((IS_PATTERN_USAGE (4) || IS_PATTERN_USAGE (5)) && Main (idx)> 0.0)
        {
            // - - perform the extended analysis of the oscillator state
            ExtState (idx);
            // - - if the model 4 is used, look for the «divergence» signal
            if (IS_PATTERN_USAGE (4) && CompareMaps (1,1)) // 0000 0001b
                result=m_pattern_4; // signal number 4
            // - - if the model 5 is used, look for the «double divergence» signal
            if (IS_PATTERN_USAGE (5) && CompareMaps (0x11,2)) // 0001 0001b
                return (m_pattern_5); // signal number 5
        }
    }
    // - - return the result
    return (result);
}
//+-----+
```

Теперь можно приступить к коду нашего индикатора.

Будем рисовать индикатор в виде линии по ценам открытия, которая будет менять цвет в зависимости от прогноза на рост или снижение цены.

```
#property indicator_chart_window
```

```
#include <Expert\ExpertInd.mqh>
```

```
#include <Expert\Signal\SignalMACDInd.mqh>
```

```
#include <Expert\Signal\SignalMAInd.mqh>
```

```
#property indicator_buffers 2
```

```
#property indicator_plots 1
```

```
#property indicator_type1 DRAW_COLOR_LINE
#property indicator_color1 clrBlack, clrRed, clrLawnGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 2

double InBuffer [];
double ColorBuffer [];
int bars_calculated=0;

input int Signal_ThresholdOpen =20; // Signal threshold value to open [0...100]
input int Signal_ThresholdClose =20; // Signal threshold value to close [0...100]

input int Signal_MACD_PeriodFast =12; // MACD (12,24,9,PRICE_CLOSE) Period
of fast EMA
input int Signal_MACD_PeriodSlow =24; // MACD (12,24,9,PRICE_CLOSE) Period
of slow EMA
input int Signal_MACD_PeriodSignal =9; // MACD (12,24,9,PRICE_CLOSE) Period
of averaging of difference
input ENUM_APPLIED_PRICE Signal_MACD_Applied =PRICE_CLOSE; // MACD
(12,24,9,PRICE_CLOSE) Prices series
input double Signal_MACD_Weight =1.0; // MACD (12,24,9,PRICE_CLOSE) Weight [0...
1.0]
input int Signal_MA_PeriodMA =12; // Moving Average (12,0,...) Period of averaging
input int Signal_MA_Shift =0; // Moving Average (12,0,...) Time shift
input ENUM_MA_METHOD Signal_MA_Method =MODE_SMA; // Moving Average
(12,0,...) Method of averaging
input ENUM_APPLIED_PRICE Signal_MA_Applied =PRICE_CLOSE; // Moving
Average (12,0,...) Prices series
input double Signal_MA_Weight =1.0; // Moving Average (12,0,...) Weight [0...1.0]

CExpertInd ExtExpert;
CSignalMAInd *filter0 = new CSignalMAInd;
CSignalMACDInd *filter1 = new CSignalMACDInd;

//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit ()
{

// - - Initializing expert
if(!ExtExpert.Init (Symbol (),Period (),true,100))
{
// - - failed
printf (__FUNCTION__+" : error initializing expert»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// - - Creating signal
```

```

CExpertSignal *signal=new CExpertSignal;
if (signal==NULL)
{
// -- failed
printf (__FUNCTION__+" : error creating signal»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- —
ExtExpert.InitSignal (signal);

filter0.PeriodMA (Signal_MA_PeriodMA);
filter0.Shift (Signal_MA_Shift);
filter0.Method (Signal_MA_Method);
filter0.Applied (Signal_MA_Applied);

filter1.PeriodFast (Signal_MACD_PeriodFast);
filter1.PeriodSlow (Signal_MACD_PeriodSlow);
filter1.PeriodSignal (Signal_MACD_PeriodSignal);
filter1.Applied (Signal_MACD_Applied);

signal.AddFilter (filter0);
signal.AddFilter (filter1);
if(!ExtExpert.ValidationSettings ())
{
// -- failed
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- Tuning of all necessary indicators
if(!ExtExpert.InitIndicators ())
{
// -- failed
printf (__FUNCTION__+" : error initializing indicators»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- ok
// -- indicator buffers mapping
SetIndexBuffer (0,InBuffer, INDICATOR_DATA);
SetIndexBuffer (1,ColorBuffer, INDICATOR_COLOR_INDEX);

ArraySetAsSeries (InBuffer, true);
ArraySetAsSeries (ColorBuffer, true);

// -- —
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 [])
{
// - - количество копируемых значений из индикатора
int values_to_copy;
// - - узнаем количество рассчитанных значений в индикаторе
int calculated=MathMin(filter0.BarsCalculatedInd (), filter1.BarsCalculatedInd ());
if (calculated <=0)
{
PrintFormat («BarsCalculated () вернул %d, код ошибки %d», calculated, GetLastError ());
return (0);
}
// - - если это первый запуск вычислений нашего индикатора или изменилось количество значений в индикаторе
// - - или если необходимо рассчитать индикатор для двух или более баров (значит что-то изменилось в истории)
if (prev_calculated==0 || calculated!=bars_calculated || rates_total> prev_calculated+1)
{
// - - если массив больше, чем значений в индикаторе на паре symbol/period, то копируем не все
// - - в противном случае копировать будем меньше, чем размер индикаторных буферов
if (calculated> rates_total) values_to_copy=rates_total;
else values_to_copy=calculated;
}
else
{
// - - значит наш индикатор рассчитывается не в первый раз и с момента последнего вызова OnCalculate ()
// - - для расчета добавилось не более одного бара
values_to_copy= (rates_total-prev_calculated) +1;
}

bars_calculated=calculated;

ArraySetAsSeries (open, true);

ExtExpert.RefreshInd ();

if (values_to_copy> 1)
```

```

{

for (int i=0; i <values_to_copy; i++) {

    ColorBuffer [i] =0;
    InBuffer [i] =open [i];

    double                                result0=Signal_MA_Weight*(filter0.LongConditionInd(i)-
filter0.ShortConditionInd (i));
    double                                result1=Signal_MACD_Weight*(filter1.LongConditionInd(i)-
filter1.ShortConditionInd (i));
    double result= (result0+result1) /2;

    if (result> =Signal_ThresholdOpen)
    {
        ColorBuffer [i] =2;
    }

    if (-result> =Signal_ThresholdOpen) {
        ColorBuffer [i] =1;
    }
    }
    }
    // - - return value of prev_calculated for next call
    return (rates_total);
    }
    //+-----+

```

Здесь мы взяли входные параметры и код инициализации из кода сгенерированного эксперта.

Так как модули сигналов работают с данными как с таймсериями, применим функцию `ArraySetAsSeries` к буферам нашего индикатора.

В функции `OnCalculate` индикатора мы сначала обновляем все данные, а затем получаем взвешенные сигналы модулей и сравниваем их с пороговым значением. В итоге получаем прогноз на увеличение или уменьшение цены.

Генетические алгоритмы

При создании самооптимизирующегося советника, на определенном этапе его работы, требуется автоматический вызов кода, который заново оптимизирует параметры советника на истории финансового инструмента, и далее советник продолжит свою работу уже с новыми параметрами.

Так как сам советник работает на текущем баре и не использует историю символа, код оптимизации параметров советника должен опираться на код индикатора, который в свою очередь создан на основе кода советника.

Для оптимизации параметров советника, или теперь уже параметров индикатора, совпадающих с параметрами советника, можно использовать полный перебор, однако для сокращения времени оптимизации можно применить генетический алгоритм.

Обратимся к терминологии.

Хромосома состоит из набора генов – набора случайно выбранных параметров советника в допустимых диапазонах.

Поклоение это набор хромосом.

Фитнес функция FF это код, возвращающий значение советника, по которому производится оптимизация, например прибыль.

Значение функции фитнеса VFF () используется для вычисления вероятности воспроизведения хромосомы – члена популяции.

$$f(x_i)$$

$$p(x_i^t) = \frac{f(x_i^t)}{\sum_{j=1}^N f(x_j^t)}$$

Начальная популяция формируется случайным образом и размер популяции (количество особей) фиксируется и не изменяется в течение работы всего алгоритма.

На основе вероятностей воспроизведения хромосом начальной популяции формируется новая популяция, и так далее пока не сработает условие завершения работы алгоритма.

Каждая следующая популяция формируется из предыдущей с помощью отбора, скрещивания и мутации.

Для отбора могут использоваться такие алгоритмы как рулетка, турнирный отбор и элитный отбор.

При применении рулетки, колесо рулетки делится на сектора, число которых совпадает с числом особей популяции. Площадь каждого сектора пропорциональна вероятности воспроизведения особи популяции. Отбор производится с помощью вращения колеса рулетки. Таким образом, особь с наибольшим значением вероятности воспроизведения попадает в следующую популяцию наибольшее число раз. Количество копий особи в следующей популяции определяется по формуле:

$$M = P(x_i) * N$$

Где N – число особей в популяции.

При турнирном отборе из популяции случайно выбираются две особи, из которых остается особь с наибольшим значением вероятности воспроизведения.

При элитном отборе несколько лучших особей переходят в следующую популяцию без изменений, не участвуя в отборе и скрещивании.

После отбора идет скрещивание особей, при котором сначала выбираются две особи, затем случайно определяется точка разрыва в диапазоне числа параметров фитнес функции, после чего особи обмениваются сегментами хромосомы. Само скрещивание производится с вероятностью $\frac{1}{2}$.

При применении мутации, сначала случайно выбирается параметр фитнес функции, затем он модифицируется. Сама мутация производится с вероятностью 0,001.

Библиотека UGAlib MQL5 Community реализации генетического алгоритма использует представление хромосомы в виде массива `double Chromosome []`, где 0 индекс это значение фитнес функции, а остальные индексы это параметры фитнес функции или гены хромосомы.

Оптимизация параметров фитнес функции ведется в одном диапазоне от `RangeMinimum` до `RangeMaximum`. Поэтому если оптимизируемые параметры имеют разные диапазоны, их нужно привести к одному диапазону, как правило, от 0.0 до 1.0.

Популяция особей представлена двумерным массивом `double Population [] [1000]`, где строки это хромосомы.

Популяция ранжируется по значению фитнес функции таким образом, что первый ее член имеет наилучшее значение фитнес функции по критерию оптимизации.

Популяция делится на две части. Вначале вся популяция заселяется особями со случайно выбранными генами в диапазоне. Затем для этих особей вычисляется значение фитнес функции, которое помещается в 0 индекс хромосомы. При этом функция GetFitness расчета значения фитнес функции оперирует колонией потомков, представленной массивом `double Colony [] [500]`, имеющим размер в два раза меньший, чем размер популяции. Таким образом, популяция заселяется двумя колониями потомков.

Колония потомков имеет размер меньший, чем размер популяции, для того, чтобы после мутаций и скрещиваний заселить ту часть популяции, которая имеет худшие значения фитнес функции. При этом особи, полученные в результате мутаций и скрещиваний, заселяют именно колонию потомков.

После начального заселения популяции производится удаление дубликатов с помощью функции `RemovalDuplicates`, в которой также производится ранжирование популяции по значению фитнес функции.

После подготовки начальной популяции вызывается цикл эпох – цикл рождения новых популяций, который продолжается до тех пор, пока количество эпох без улучшения не превысит порог `EPOCH`.

Критерием улучшения служит эталонная хромосома – первая особь популяции.

В цикле эпох популяция модифицируется с помощью репликации, естественной мутации, искусственной мутации, заимствования генов и скрещивания, применяемых для заселения новой колонии потомков, которая затем замещает часть популяции, имеющей худшие значения фитнес функции.

В цикле заселения новой колонии потомков функции `CycleOfOperators`, операторы репликации, естественной мутации, искусственной мутации, заимствования генов и скрещивания выбираются случайно, в зависимости от их доли от 0 до 100.

После создания новой популяции, в ней также удаляются дубликаты, и она ранжируется по значению фитнес функции.

Здесь репликация заключается в выборе двух хромосом популяции и создании на их основе новой хромосомы, для которой гены случайно выбираются в расширенном диапазоне с помощью сдвига от гена первой особи влево и от гена второй особи вправо.

Выбор двух родителей из популяции осуществляется с помощью алгоритма рулетки, упомянутого выше.

Естественная мутация производится с помощью выбора одного родителя из популяции, используя алгоритм рулетки, и замены его генов генами, случайно выбранными в диапазоне от `RangeMinimum` до `RangeMaximum`. Замена генов производится с вероятностью `NMutationProbability` от 0.0 до 100.0.

При искусственной мутации выбираются два родителя из популяции, используя алгоритм рулетки, и гены потомка случайно выбираются из незанятого генами родителей пространства на числовой прямой в диапазонах от `RangeMinimum` до сдвига от гена первой особи вправо и от сдвига от гена второй особи влево до `RangeMaximum`.

При заимствовании генов для первого гена потомка выбирается родитель из популяции, используя алгоритм рулетки, и берется у него первый ген, далее, для второго гена отбирается второй родитель и берется второй ген и т. д.

При скрещивании выбираются два родителя из популяции, используя алгоритм рулетки, и гены потомка формируются за счет обмена отрезками хромосом мамы и папы.

Полный код реализации генетического алгоритма.

```
//+-----+  
//| JQS UGA v1.3.1 |
```

```
//| Copyright © 2010, JQS aka Joo. |
//| http://www.mql4.com/ru/users/joo |
//| https://login.mql5.com/ru/users/joo |
//+-----+
//Библиотека «Универсального Генетического Алгоритма UGAlib» |
//использующего представление хромосомы вещественными числами. |
//+-----+

//----- Глобальные переменные -----
double Chromosome []; //Набор оптимизируемых аргументов функции – генов
// (например: веса нейронной сети и т.д.) -хромосома
int ChromosomeCount=0; //Максимально возможное количество хромосом в колонии
int TotalOfChromosomesInHistory=0; //Общее количество хромосом в истории
int ChrCountInHistory=0; //Количество уникальных хромосом в базе хромосом
int GeneCount=0; //Количество генов в хромосоме

double RangeMinimum=0.0; //Минимум диапазона поиска
double RangeMaximum=0.0; //Максимум диапазона поиска
double Precision=0.0; //Шаг поиска
int OptimizeMethod=0; //1-минимум, любое другое – максимум

double Population [] [1000]; //Популяция
double Colony [] [500]; //Колония потомков
int PopulChromosCount=0; //Текущее количество хромосом в популяции
int Epoch=0; //Кол-во эпох без улучшения
int AmountStartsFF=0; //Количество запусков функции приспособленности
//-----

//-----
//Основная функция UGA
void UGA
(
double ReplicationPortion, //Доля Репликации.
double NMutationPortion, //Доля Естественной мутации.
double ArtificialMutation, //Доля Искусственной мутации.
double GenoMergingPortion, //Доля Заимствования генов.
double CrossingOverPortion, //Доля Кроссинговера.
//-----
double ReplicationOffset, //Коэффициент смещения границ интервала
double NMutationProbability, //Вероятность мутации каждого гена в %
)
{
//сброс генератора, производится только один раз
MathSrand ((int) TimeLocal ());
//----- -Переменные -----
int chromos=0, gene=0; //индексы хромосом и генов
int resetCounterFF=0; //счетчик сбросов «Эпох без улучшений»
int currentEpoch=1; //номер текущей эпохи
int SumOfCurrentEpoch=0; //сумма «Эпох без улучшений»
```

```
int MinOfCurrentEpoch=Epoch;//минимальное «Эпох без улучшений»
int MaxOfCurrentEpoch=0;//максимальное «Эпох без улучшений»
int epochGlob =0;//общее количество эпох
// Колония [количество признаков (генов)] [количество особей в колонии]
ArrayResize (Population, GeneCount+1);
ArrayInitialize (Population,0.0);
// Колония потомков [количество признаков (генов)] [количество особей в колонии]
ArrayResize (Colony, GeneCount+1);
ArrayInitialize (Colony,0.0);
// Банк хромосом
// [количество признаков (генов)] [количество хромосом в банке]
double historyHromosomes [] [100000];
ArrayResize (historyHromosomes, GeneCount+1);
ArrayInitialize (historyHromosomes,0.0);
// -----
// ----- Проверка корректности входных параметров -----
// ...количество хромосом должно быть не меньше 2
if (ChromosomeCount <=1) ChromosomeCount=2;
if (ChromosomeCount> 500) ChromosomeCount=500;
// -----
//
```

```
// 1) Создать протопопуляцию -----1)
ProtopopulationBuilding ();
//
```

```
// 2) Определить приспособленность каждой особи -----2)
//Для 1-ой колонии
for (chromos=0;chromos <ChromosomeCount; chromos++)
for (gene=1;gene <=GeneCount; gene++)
Colony [gene] [chromos] =Population [gene] [chromos];

GetFitness (historyHromosomes);

for (chromos=0;chromos <ChromosomeCount; chromos++)
Population [0] [chromos] =Colony [0] [chromos];

//Для 2-ой колонии
for (chromos=ChromosomeCount; chromos <ChromosomeCount*2;chromos++)
for (gene=1;gene <=GeneCount; gene++)
Colony [gene] [chromos-ChromosomeCount] =Population [gene] [chromos];

GetFitness (historyHromosomes);

for (chromos=ChromosomeCount; chromos <ChromosomeCount*2;chromos++)
Population [0] [chromos] =Colony [0] [chromos-ChromosomeCount];
//
```

```
// 3) Подготовить популяцию к размножению -----3)
```

```
RemovalDuplicates ();  
//
```

```
// 4) Выделить эталонную хромосому — — — —4)  
for (gene=0;gene <=GeneCount; gene++)  
Chromosome [gene] =Population [gene] [0];  
//
```

```
ServiceFunction ();
```

```
//Основной цикл генетического алгоритма с 5 по 6  
while (currentEpoch <=Epoch)  
{  
//
```

```
// 5) Операторы UGA — — — —5)  
CycleOfOperators  
(  
historyHromosomes,  
// — —  
ReplicationPortion, //Доля Репликации.  
NMutationPortion, //Доля Естественной мутации.  
ArtificialMutation, //Доля Искусственной мутации.  
GenoMergingPortion, //Доля Заимствования генов.  
CrossingOverPortion, //Доля Кроссинговера.  
// — —  
ReplicationOffset, //Коэффициент смещения границ интервала  
NMutationProbability//Вероятность мутации каждого гена в %  
);  
//
```

```
// 6) Сравнить гены лучшего потомка с генами эталонной хромосомы.  
// Если хромосома лучшего потомка лучше эталонной,  
// заменить эталонную. — — — —6)  
//Если режим оптимизации – минимизация  
if (OptimizeMethod==1)  
{  
//Если лучшая хромосома популяции лучше эталонной  
if (Population [0] [0] <Chromosome [0])  
{  
//Заменим эталонную хромосому  
for (gene=0;gene <=GeneCount; gene++)  
Chromosome [gene] =Population [gene] [0];  
ServiceFunction ();  
//Сбросим счетчик «эпох без улучшений»  
if (currentEpoch <MinOfCurrentEpoch)  
MinOfCurrentEpoch=currentEpoch;  
if (currentEpoch> MaxOfCurrentEpoch)  
MaxOfCurrentEpoch=currentEpoch;
```

```
SumOfCurrentEpoch+=currentEpoch; currentEpoch=1; resetCounterFF++;
}
else
currentEpoch++;
}
//Если режим оптимизации – максимизация
else
{
//Если лучшая хромосома популяции лучше эталонной
if (Population [0] [0]> Chromosome [0])
{
//Заменяем эталонную хромосому
for (gene=0;gene <=GeneCount; gene++)
Chromosome [gene] =Population [gene] [0];
ServiceFunction ();
//Сбросим счетчик «эпох без улучшений»
if (currentEpoch <MinOfCurrentEpoch)
MinOfCurrentEpoch=currentEpoch;
if (currentEpoch> MaxOfCurrentEpoch)
MaxOfCurrentEpoch=currentEpoch;
SumOfCurrentEpoch+=currentEpoch; currentEpoch=1; resetCounterFF++;
}
else
currentEpoch++;
}
//
=====

//Прошла ещё одна эпоха...
epochGlob++;
}
Print («Прошло всего эпох=», epochGlob,» Всего сбросов=», resetCounterFF);
Print («Мин. эпох без улучш.=», MinOfCurrentEpoch,
«Средн. эпох без улучш.=»,
NormalizeDouble ((double) SumOfCurrentEpoch/ (double) resetCounterFF,2),
«Макс. эпох без улучш.=», MaxOfCurrentEpoch);
Print (ChrCountInHistory,» – Уникальных хромосом»);
Print (AmountStartsFF,» – Общее кол-во запусков FF»);
Print (TotalOfChromosomesInHistory,» – Общее кол-во хромосом в истории»);
Print (NormalizeDouble (100.0- ((double) ChrCountInHistory*100.0/
(double) TotalOfChromosomesInHistory),2),»% дубликатов»);
Print (Chromosome [0],» – Лучший результат»);
}
// -----

// -----

//Создание протопопуляции
void ProtopopulationBuilding ()
{
PopulChromosCount=ChromosomeCount*2;
```

```
//Заполнить популяцию хромосомами со случайными
//...генами в диапазоне RangeMinimum... RangeMaximum
for (int chromos=0;chromos <PopulChromosCount; chromos++)
{
//начиная с 1-го индекса (0-ой -зарезервирован для VFF)
for (int gene=1;gene <=GeneCount; gene++)
Population [gene] [chromos] =
SelectInDiscreteSpace (RNDfromCI (RangeMinimum, RangeMaximum),RangeMinimum,
RangeMaximum, Precision,3);
TotalOfChromosomesInHistory++;
}
}
// -----

// -----
//Получение приспособленности для каждой особи.
void GetFitness
(
double &historyHromosomes [] [100000]
)
{
for (int chromos=0;chromos <ChromosomeCount; chromos++)
CheckHistoryChromosomes (chromos, historyHromosomes);
}
// -----

// -----
//Проверка хромосомы по базе хромосом.
void CheckHistoryChromosomes
(
int chromos,
double &historyHromosomes [] [100000]
)
{
// ----- -Переменные -----
int Ch1=0; //Индекс хромосомы из базы
int Ge =0; //Индекс гена
int cnt=0; //Счетчик уникальных генов. Если хоть один ген отличается
//– хромосома признается уникальной
// -----

//Если в базе хранится хоть одна хромосома
if (ChrCountInHistory> 0)
{
//Переберем хромосомы в базе, чтобы найти такую же
for (Ch1=0;Ch1 <ChrCountInHistory && cnt <GeneCount; Ch1++)
{
cnt=0;
//Сверяем гены, пока индекс гена меньше кол-ва генов и пока попадают одинако-
вые гены
```

```
for (Ge=1;Ge <=GeneCount; Ge++)
{
if (Colony [Ge] [chromos]!=historyHromosomes [Ge] [Ch1])
break;
cnt++;
}
}
//Если набралось одинаковых генов столько же, можно взять готовое решение из базы
if (cnt==GeneCount)
Colony [0] [chromos] =historyHromosomes [0] [Ch1—1];
//Если нет такой же хромосомы в базе, то рассчитаем для неё FF...
else
{
FitnessFunction (chromos);
//.. и если есть место в базе сохраним
if (ChrCountInHistory <100000)
{
for (Ge=0;Ge <=GeneCount; Ge++)
historyHromosomes [Ge] [ChrCountInHistory] =Colony [Ge] [chromos];
ChrCountInHistory++;
}
}
}
//Если база пустая, рассчитаем для неё FF и сохраним её в базе
else
{
FitnessFunction (chromos);
for (Ge=0;Ge <=GeneCount; Ge++)
historyHromosomes [Ge] [ChrCountInHistory] =Colony [Ge] [chromos];
ChrCountInHistory++;
}
}
// -----

// -----
//Цикл операторов UGA
void CycleOfOperators
(
double &historyHromosomes [] [100000],
// — —
double ReplicationPortion, //Доля Репликации.
double NMutationPortion, //Доля Естественной мутации.
double ArtificialMutation, //Доля Искусственной мутации.
double GenoMergingPortion, //Доля Заимствования генов.
double CrossingOverPortion, //Доля Кроссинговера.
// — —
double ReplicationOffset, //Коэффициент смещения границ интервала
double NMutationProbability //Вероятность мутации каждого гена в %
)
```

```
{
// ----- -Переменные-----
double child [];
ArrayResize (child, GeneCount+1);
ArrayInitialize (child,0.0);

int gene=0,chromos=0, border=0;
int i=0,u=0;
double p=0.0,start=0.0;
double fit [] [2];
ArrayResize (fit,6);
ArrayInitialize (fit,0.0);

//Счетчик посадочных мест в новой популяции.
int T=0;
// -----

//Зададим долю операторов UGA
double portion [6];
portion [0] =ReplicationPortion; //Доля Репликации.
portion [1] =NMutationPortion; //Доля Естественной мутации.
portion [2] =ArtificialMutation; //Доля Искусственной мутации.
portion [3] =GenoMergingPortion; //Доля Заимствования генов.
portion [4] =CrossingOverPortion; //Доля Кроссинговера.
portion [5] =0.0;
// -----
if (NMutationProbability <0.0)
NMutationProbability=0.0;
if (NMutationProbability> 100.0)
NMutationProbability=100.0;
// -----
// ----- Цикл операторов UGA -----
//Заполняем новую колонию потомками
while (T <ChromosomeCount)
{
//=====

for (i=0;i <6;i++)
{
fit [i] [0] =start;
fit [i] [1] =start+MathAbs (portion [i] -portion [5]);
start=fit [i] [1];

}

p=RNDfromCI (fit [0] [0],fit [4] [1]);
for (u=0;u <5;u++)
{
if ((fit [u] [0] <=p && p <fit [u] [1]) || p==fit [u] [1])
```

```
break;
}
Print (u);
//=====================================================
switch (u)
{
// -----
case 0:
// ----- Репликация -----
//Если есть место в новой колонии, создадим новую особь
if (T <ChromosomeCount)
{
Replication (child, ReplicationOffset);
//Поселим новую особь в новую колонию
for (gene=1;gene <=GeneCount; gene++) Colony [gene] [T] =child [gene];
//Одно место заняли, счетчик перемотаем вперед
T++;
TotalOfChromosomesInHistory++;
}
// -----
break;
// -----
case 1:
// ----- Естественная мутация -----
//Если есть место в новой колонии, создадим новую особь
if (T <ChromosomeCount)
{
NaturalMutation (child, NMutationProbability);
//Поселим новую особь в новую колонию
for (gene=1;gene <=GeneCount; gene++) Colony [gene] [T] =child [gene];
//Одно место заняли, счетчик перемотаем вперед
T++;
TotalOfChromosomesInHistory++;
}
// -----
break;
// -----
case 2:
// ----- Искусственная мутация -----
//Если есть место в новой колонии, создадим новую особь
if (T <ChromosomeCount)
{
ArtificialMutation (child, ReplicationOffset);
//Поселим новую особь в новую колонию
for (gene=1;gene <=GeneCount; gene++) Colony [gene] [T] =child [gene];
//Одно место заняли, счетчик перемотаем вперед
T++;
TotalOfChromosomesInHistory++;
}
}
```

```
// -----  
break;  
// -----  
case 3:  
// ----- -Образование особи с заимствованными генами -----  
//Если есть место в новой колонии, создадим новую особь  
if (T <ChromosomeCount)  
{  
    GenoMerging (child);  
    //Поселим новую особь в новую колонию  
    for (gene=1;gene <=GeneCount; gene++) Colony [gene] [T] =child [gene];  
    //Одно место заняли, счетчик перемотаем вперед  
    T++;  
    TotalOfChromosomesInHistory++;  
}  
// -----  
break;  
// -----  
default:  
// ----- -Кроссинговер -----  
//Если есть место в новой колонии, создадим новую особь  
if (T <ChromosomeCount)  
{  
    CrossingOver (child);  
    //Поселим новую особь в новую колонию  
    for (gene=1;gene <=GeneCount; gene++) Colony [gene] [T] =child [gene];  
    //Одно место заняли, счетчик перемотаем вперед  
    T++;  
    TotalOfChromosomesInHistory++;  
}  
// -----  
  
break;  
// -----  
}  
} //Конец цикла операторов UGA —  
  
//Определим приспособленность каждой особи в колонии потомков  
GetFitness (historyHromosomes);  
  
//Поселим потомков в основную популяцию  
if (PopulChromosCount> =ChromosomeCount)  
{  
    border=ChromosomeCount;  
    PopulChromosCount=ChromosomeCount*2;  
}  
else  
{  
    border=PopulChromosCount;
```

```
PopulChromosCount+=ChromosomeCount;
}
for (chromos=0;chromos <ChromosomeCount; chromos++)
for (gene=0;gene <=GeneCount; gene++)
Population [gene] [chromos+border] =Colony [gene] [chromos];
```

```
//Подготовим популяцию к следующему размножению
RemovalDuplicates ();
} //конец ф-ии
// -----
```

```
// -----
//Репликация
void Replication
(
double &child [],
double ReplicationOffset
)
{
// ----- -Переменные -----
double C1=0.0,C2=0.0,temp=0.0,Maximum=0.0,Minimum=0.0;
int address_mama=0,address_papa=0;
// -----
SelectTwoParents (address_mama, address_papa);
// ----- -Цикл перебора генов -----
for (int i=1;i <=GeneCount; i++)
{
// --- определим откуда мать и отец ---
C1 = Population [i] [address_mama];
C2 = Population [i] [address_papa];
// -----
```

```
// -----
//...определим наибольший и наименьший из них,
//если C1> C2, поменяем их местами
if (C1> C2)
{
temp = C1; C1=C2; C2 = temp;
}
// -----
if (C2-C1 <Precision)
{
child [i] =C1; continue;
}
// -----
//Назначим границы создания нового гена
Minimum = C1- ((C2-C1) *ReplicationOffset);
Maximum = C2+ ((C2-C1) *ReplicationOffset);
// -----
```

```
//Обязательная проверка, что бы поиск не вышел из заданного диапазона
if (Minimum < RangeMinimum) Minimum = RangeMinimum;
if (Maximum > RangeMaximum) Maximum = RangeMaximum;
// -----
temp=RNDfromCI (Minimum, Maximum);
child [i] =
SelectInDiscreteSpace (temp, RangeMinimum, RangeMaximum, Precision,3);
}
}
// -----

// -----
//Естественная мутация.
void NaturalMutation
(
double &child [],
double NMutationProbability
)
{
// ----- -Переменные -----
int address=0;
// -----

// ----- -Отбор родителя -----
SelectOneParent (address);
// -----
for (int i=1;i <=GeneCount; i++)
if (RNDfromCI (0.0,100.0) <=NMutationProbability)
child [i] =
SelectInDiscreteSpace (RNDfromCI (RangeMinimum, RangeMaximum),RangeMinimum,
RangeMaximum, Precision,3);
else
child [i] =Population [i] [address];
}
// -----

// -----
//Искусственная мутация.
void ArtificialMutation
(
double &child [],
double ReplicationOffset
)
{
// ----- -Переменные -----
double C1=0.0,C2=0.0,temp=0.0,Maximum=0.0,Minimum=0.0,p=0.0;
int address_mama=0,address_papa=0;
// -----
// ----- -Отбор родителей -----
```

```
SelectTwoParents (address_mama, address_papa);
// -----
// ----- -Цикл перебора генов -----
for (int i=1;i <=GeneCount; i++)
{
// --- определим откуда мать и отец ---
C1 = Population [i] [address_mama];
C2 = Population [i] [address_papa];
// -----

// -----
//...определим наибольший и наименьший из них,
//если C1> C2, поменяем их местами
if (C1> C2)
{
temp=C1; C1=C2; C2=temp;
}
// -----
//Назначим границы создания нового гена
Minimum=C1- ((C2-C1) *ReplicationOffset);
Maximum=C2+ ((C2-C1) *ReplicationOffset);
// -----
//Обязательная проверка, что бы поиск не вышел из заданного диапазона
if (Minimum <RangeMinimum) Minimum = RangeMinimum;
if (Maximum> RangeMaximum) Maximum = RangeMaximum;
// -----
p=MathRand ();
if (p <16383.5)
{
temp=RNDfromCI (RangeMinimum, Minimum);
child [i] =
SelectInDiscreteSpace (temp, RangeMinimum, RangeMaximum, Precision,3);
}
else
{
temp=RNDfromCI (Maximum, RangeMaximum);
child [i] =
SelectInDiscreteSpace (temp, RangeMinimum, RangeMaximum, Precision,3);
}
}
}
// -----

// -----
//Заимствование генов.
void GenoMerging
(
double &child []
)
```

```
{
// ----- -Переменные -----
int address=0;
// -----
for (int i=1;i <=GeneCount; i++)
{
// ----- -Отбор родителя -----
SelectOneParent (address);
// -----
child [i] =Population [i] [address];
}
}
// -----

// -----
//Кроссинговер.
void CrossingOver
(
double &child []
)
{
// ----- -Переменные -----
int address_mama=0,address_papa=0;
// -----
// ----- -Отбор родителей -----
SelectTwoParents (address_mama, address_papa);
// -----
//Определим точку разрыва
int address_of_gene= (int) MathFloor ((GeneCount-1) * (MathRand () /32767.5));

for (int i=1;i <=GeneCount; i++)
{
// ----- копируем гены матери -----
if (i <=address_of_gene+1)
child [i] =Population [i] [address_mama];
// ----- копируем гены отца -----
else
child [i] =Population [i] [address_papa];
}
}
// -----

// -----
//Отбор двух родителей.
void SelectTwoParents
(
int &address_mama,
int &address_papa
)
```

```
{
// ----- -Переменные -----
int cnt=1;
address_mama=0;//адрес материнской особи в популяции
address_papa=0;//адрес отцовской особи в популяции
// -----
// ----- Отбор родителей -----
//Десять попыток выбрать разных родителей.
while (cnt <=10)
{
//Для материнской особи
address_mama=NaturalSelection ();
//Для отцовской особи
address_papa=NaturalSelection ();
if (address_mama!=address_papa)
break;
cnt++;
}
// -----
}
// -----

// -----
//Отбор одного родителя.
void SelectOneParent
(
int &address//адрес родительской особи в популяции
)
{
// ----- -Переменные -----
address=0;
// -----
// ----- Отбор родителя -----
address=NaturalSelection ();
// -----
}
// -----

// -----
//Естественный отбор.
int NaturalSelection ()
{
// ----- -Переменные -----
int i=0,u=0;
double p=0.0,start=0.0;
double fit [] [2];
ArrayResize (fit, PopulChromosCount);
ArrayInitialize (fit,0.0);
```

```
double delta= (Population [0] [0] -Population [0] [PopulChromosCount-1]) *0.01-Population
[0] [PopulChromosCount-1];
// -----

for (i=0;i <PopulChromosCount; i++)
{
fit [i] [0] =start;
fit [i] [1] =start+MathAbs (Population [0] [i] +delta);
start=fit [i] [1];
}
p=RNDfromCI (fit [0] [0],fit [PopulChromosCount-1] [1]);

for (u=0;u <PopulChromosCount; u++)
if ((fit [u] [0] <=p && p <fit [u] [1]) || p==fit [u] [1])
break;

return (u);
}
// -----

// -----
//Удаление дубликатов с сортировкой по VFF
void RemovalDuplicates ()
{
// ----- -Переменные -----
int chromosomeUnique [1000];//Массив хранит признак уникальности
//каждой хромосомы: 0-дубликат, 1-уникальная
ArrayInitialize (chromosomeUnique,1); //Предположим, что дубликатов нет
double PopulationTemp [] [1000];
ArrayResize (PopulationTemp, GeneCount+1);
ArrayInitialize (PopulationTemp,0.0);

int Ge =0; //Индекс гена
int Ch =0; //Индекс хромосомы
int Ch2=0; //Индекс второй хромосомы
int cnt=0; //Счетчик
// -----

// ----- Удалим дубликаты ----- -1
//Выбираем первый из пары для сравнения...
for (Ch=0;Ch <PopulChromosCount-1;Ch++)
{
//Если не дубликат...
if (chromosomeUnique [Ch]!=0)
{
//Выбираем второй из пары...
for (Ch2=Ch+1;Ch2 <PopulChromosCount; Ch2++)
{
if (chromosomeUnique [Ch2]!=0)
```

```
{
//Обнулим счетчик количества идентичных генов
cnt=0;
//Сверяем гены, пока попадают одинаковые гены
for (Ge=1;Ge <=GeneCount; Ge++)
{
if (Population [Ge] [Ch]!=Population [Ge] [Ch2])
break;
else
cnt++;
}
//Если набралось одинаковых генов столько же, сколько всего генов
//..хромосома признается дубликатом
if (cnt==GeneCount)
chromosomeUnique [Ch2] =0;
}
}
}
//Счетчик посчитает количество уникальных хромосом
cnt=0;
//Скопируем уникальные хромосомы во временный массив
for (Ch=0;Ch <PopulChromosCount; Ch++)
{
//Если хромосома уникальна, скопируем её, если нет, перейдем к следующей
if (chromosomeUnique [Ch] ==1)
{
for (Ge=0;Ge <=GeneCount; Ge++)
PopulationTemp [Ge] [cnt] =Population [Ge] [Ch];
cnt++;
}
}
//Назначим переменной «Всего хромосом» значение счетчика уникальных хромосом
PopulChromosCount=cnt;
//Вернем уникальные хромосомы обратно в массив для временного хранения
//..объединяемых популяций
for (Ch=0;Ch <PopulChromosCount; Ch++)
for (Ge=0;Ge <=GeneCount; Ge++)
Population [Ge] [Ch] =PopulationTemp [Ge] [Ch];
//
```

=====1

```
// — — — — — Ранжирование популяции — — — — — -2
PopulationRanking ();
//
```

=====2

```
}
// — — — — —
```

```
// -----  
//Ранжирование популяции.  
void PopulationRanking ()  
{  
// ----- -Переменные -----  
int cnt=1, i = 0, u = 0;  
double PopulationTemp [] [1000]; //Временная популяция  
ArrayResize (PopulationTemp, GeneCount+1);  
ArrayInitialize (PopulationTemp,0.0);  
  
int Indexes []; //Индексы хромосом  
ArrayResize (Indexes, PopulChromosCount);  
ArrayInitialize (Indexes,0);  
int t0=0;  
double ValueOnIndexes []; //VFF соответствующих  
//..индексов хромосом  
ArrayResize (ValueOnIndexes, PopulChromosCount);  
ArrayInitialize (ValueOnIndexes,0.0); double t1=0.0;  
// -----  
  
//Проставим индексы во временном массиве temp2 и  
//...скопируем первую строку из сортируемого массива  
for (i=0;i <PopulChromosCount; i++)  
{  
Indexes [i] = i;  
ValueOnIndexes [i] = Population [0] [i];  
}  
if (OptimizeMethod==1)  
{  
while (cnt> 0)  
{  
cnt=0;  
for (i=0;i <PopulChromosCount-1;i++)  
{  
if (ValueOnIndexes [i]> ValueOnIndexes [i+1])  
{  
// -----  
t0 = Indexes [i+1];  
t1 = ValueOnIndexes [i+1];  
Indexes [i+1] = Indexes [i];  
ValueOnIndexes [i+1] = ValueOnIndexes [i];  
Indexes [i] = t0;  
ValueOnIndexes [i] = t1;  
// -----  
cnt++;  
}  
}  
}  
}
```

```
else
{
while (cnt> 0)
{
cnt=0;
for (i=0;i <PopulChromosCount-1;i++)
{
if (ValueOnIndexes [i] <ValueOnIndexes [i+1])
{
// -----
t0 = Indexes [i+1];
t1 = ValueOnIndexes [i+1];
Indexes [i+1] = Indexes [i];
ValueOnIndexes [i+1] = ValueOnIndexes [i];
Indexes [i] = t0;
ValueOnIndexes [i] = t1;
// -----
cnt++;
}
}
}
}
//Создадим отсортированный массив по полученным индексам
for (i=0;i <GeneCount+1;i++)
for (u=0;u <PopulChromosCount; u++)
PopulationTemp [i] [u] =Population [i] [Indexes [u]];
//Скопируем отсортированный массив обратно
for (i=0;i <GeneCount+1;i++)
for (u=0;u <PopulChromosCount; u++)
Population [i] [u] =PopulationTemp [i] [u];
}
// -----

// -----
//Генератор случайных чисел из заданного интервала.
double RNDfromCI (double Minimum, double Maximum)
{return (Minimum+ ((Maximum-Minimum) *MathRand () /32767.5));}
// -----

// -----
//Выбор в дискретном пространстве.
//Режимы:
//1-ближайшее снизу
//2-ближайшее сверху
//любое-до ближайшего
double SelectInDiscreteSpace
(
double In,
double InMin,
```

```
double InMax,
double step,
int RoundMode
)
{
if (step==0.0)
return (In);
// обеспечим правильность границ
if (InMax <InMin)
{
double temp = InMax; InMax = InMin; InMin = temp;
}
// при нарушении – вернем нарушенную границу
if (In <InMin) return (InMin);
if (In> InMax) return (InMax);
if (InMax == InMin || step <= 0.0) return (InMin);
// приведем к заданному масштабу
step = (InMax – InMin) / MathCeil ((InMax – InMin) / step);
switch (RoundMode)
{
case 1: return (InMin + step * MathFloor ((In – InMin) / step));
case 2: return (InMin + step * MathCeil ((In – InMin) / step));
default: return (InMin + step * MathRound ((In – InMin) / step));
}
}
// -----
```

Для использования библиотеки UGAlib необходимо написать две функции FitnessFunction и ServiceFunction.

Функция FitnessFunction получает на вход индекс хромосомы и рассчитывает для нее значение, по которому ведется оптимизация генов хромосомы.

Функция ServiceFunction может выводить значение фитнес функции и остальные гены эталонной хромосомы при каждом проходе оптимизации.

В качестве примера рассмотрим оптимизацию параметров индикатора, созданного в главе «Создание индикатора на основе модулей торговых сигналов эксперта».

Модифицируем код индикатора таким образом, чтобы рассчитывать виртуальные сделки на покупку и продажу финансового инструмента по сигналам индикатора.

```
#property indicator_chart_window

#include <Expert\ExpertInd.mqh>
#include <Expert\Signal\MySignal\SignalMACDExInd.mqh>
#include <Expert\Signal\MySignal\SignalMAExInd.mqh>

#property indicator_buffers 2
#property indicator_plots 1
#property indicator_type1 DRAW_COLOR_LINE
#property indicator_color1 clrBlack, clrRed, clrLawnGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 2
```

```

double InBuffer [];
double ColorBuffer [];
int bars_calculated=0;

input int Signal_ThresholdOpen =20; // Signal threshold value to open [0...100]
input int Signal_ThresholdClose =20; // Signal threshold value to close [0...100]

input int Signal_MACD_PeriodFast =12; // MACD (12,24,9,PRICE_CLOSE) Period
of fast EMA
input int Signal_MACD_PeriodSlow =24; // MACD (12,24,9,PRICE_CLOSE) Period
of slow EMA
input int Signal_MACD_PeriodSignal =9; // MACD (12,24,9,PRICE_CLOSE) Period
of averaging of difference
input ENUM_APPLIED_PRICE Signal_MACD_Applied =PRICE_CLOSE; // MACD
(12,24,9,PRICE_CLOSE) Prices series
input double Signal_MACD_Weight =1.0; // MACD (12,24,9,PRICE_CLOSE) Weight [0...
1.0]
input int Signal_MA_PeriodMA =12; // Moving Average (12,0,...) Period of averaging
input int Signal_MA_Shift =0; // Moving Average (12,0,...) Time shift
input ENUM_MA_METHOD Signal_MA_Method =MODE_SMA; // Moving Average
(12,0,...) Method of averaging
input ENUM_APPLIED_PRICE Signal_MA_Applied =PRICE_CLOSE; // Moving
Average (12,0,...) Prices series
input double Signal_MA_Weight =1.0; // Moving Average (12,0,...) Weight [0...1.0]

CExpertInd ExtExpert;
CSignalMAInd *filter0 = new CSignalMAInd;
CSignalMACDInd *filter1 = new CSignalMACDInd;

//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit ()
{
    // - - Initializing expert
    if(!ExtExpert.Init (Symbol (),Period (),true,100))
    {
        // - - failed
        printf (__FUNCTION__+": error initializing expert»);
        ExtExpert.Deinit ();
        return (INIT_FAILED);
    }
    // - - Creating signal
    CExpertSignal *signal=new CExpertSignal;
    if (signal==NULL)
    {
        // - - failed
        printf (__FUNCTION__+": error creating signal»);
    }
}

```

```
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// ————

ExtExpert.InitSignal (signal);

filter0.PeriodMA (Signal_MA_PeriodMA);
filter0.Shift (Signal_MA_Shift);
filter0.Method (Signal_MA_Method);
filter0.Applied (Signal_MA_Applied);

filter1.PeriodFast (Signal_MACD_PeriodFast);
filter1.PeriodSlow (Signal_MACD_PeriodSlow);
filter1.PeriodSignal (Signal_MACD_PeriodSignal);
filter1.Applied (Signal_MACD_Applied);

signal.AddFilter (filter0);
signal.AddFilter (filter1);
if(!ExtExpert.ValidationSettings ())
{
// - failed
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// - Tuning of all necessary indicators
if(!ExtExpert.InitIndicators ())
{
// - failed
printf (__FUNCTION__+" : error initializing indicators»");
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// - ok
// - indicator buffers mapping
SetIndexBuffer (0,InBuffer, INDICATOR_DATA);
SetIndexBuffer (1,ColorBuffer, INDICATOR_COLOR_INDEX);

ArraySetAsSeries (InBuffer, true);
ArraySetAsSeries (ColorBuffer, true);

// ————
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 [])
{
// - - количество копируемых значений из индикатора
int values_to_copy;
// - - узнаем количество рассчитанных значений в индикаторе
int calculated=MathMin(filter0.BarsCalculatedInd (), filter1.BarsCalculatedInd ());
if (calculated <=0)
{
PrintFormat («BarsCalculated () вернул %d, код ошибки %d», calculated, GetLastError ());
return (0);
}
// - - если это первый запуск вычислений нашего индикатора или изменилось количество значений в индикаторе
// - - или если необходимо рассчитать индикатор для двух или более баров (значит что-то изменилось в истории)
if (prev_calculated==0 || calculated!=bars_calculated || rates_total> prev_calculated+1)
{
// - - если массив больше, чем значений в индикаторе на паре symbol/period, то копируем не все
// - - в противном случае копировать будем меньше, чем размер индикаторных буферов
if (calculated> rates_total) values_to_copy=rates_total;
else values_to_copy=calculated;
}
else
{
// - - значит наш индикатор рассчитывается не в первый раз и с момента последнего вызова OnCalculate ()
// - - для расчета добавилось не более одного бара
values_to_copy= (rates_total-prev_calculated) +1;
}

bars_calculated=calculated;

ArraySetAsSeries (open, true);

if (values_to_copy> 1)
{

ExtExpert.RefreshInd ();

// - - флаг покупки
bool flagBuy=false;
// - флаг продажи
```

```
bool flagSell=false;
// – покупка
double priceBuy=0;
double priceStopBuy=0;
double profitBuy=0;
double profitTotalBuy=0;
int countProfitBuyPlus=0;
int countProfitBuyMinus=0;
// – продажа
double priceSell=0;
double priceStopSell=0;
double profitSell=0;
double profitTotalSell=0;
int countProfitSellPlus=0;
int countProfitSellMinus=0;
// – spread
double sp=0.0002;

for (int i=0; i < values_to_copy; i++) {

    ColorBuffer [i] =0;
    InBuffer [i] =open [i];

    double                                     result0=Signal_MA_Weight*(filter0.LongConditionInd(i)-
filter0.ShortConditionInd (i));
    double                                     result1=Signal_MACD_Weight*(filter1.LongConditionInd(i)-
filter1.ShortConditionInd (i));
    double result= (result0+result1) /2;

    if (result> =Signal_ThresholdOpen)
    {
        ColorBuffer [i] =2;
        if (flagSell==true) {
            flagSell=false;
            priceStopSell=InBuffer [i];
            profitSell= (priceStopSell-priceSell-sp) *10000;
            if (profitSell> 0) {countProfitSellPlus++;} else {countProfitSellMinus++;}
            profitTotalSell=profitTotalSell+profitSell;
        }
        if (flagBuy==false) {
            flagBuy=true;
            priceBuy=InBuffer [i];
        }
    }

    if (-result> =Signal_ThresholdOpen) {
        ColorBuffer [i] =1;
        if (flagBuy==true) {
            flagBuy=false;
```

```
priceStopBuy=InBuffer [i];
profitBuy= (priceStopBuy-priceBuy-sp) *10000;
if (profitBuy> 0) {countProfitBuyPlus++;} else {countProfitBuyMinus++;}
profitTotalBuy=profitTotalBuy+profitBuy;
}
if (flagSell==false) {
priceSell=InBuffer [i];
flagSell=true;
}
}
}
//Print (« ProfitBuy», profitTotalBuy,» countProfitBuyPlus», countProfitBuyPlus,»
countProfitBuyMinus», countProfitBuyMinus);
//Print (« ProfitSell», profitTotalSell,» countProfitSellPlus», countProfitSellPlus,»
countProfitSellMinus», countProfitSellMinus);

}
// - - return value of prev_calculated for next call
return (rates_total);
}
//+-----+
Напишем фитнес функцию на основе этого индикатора. Оптимизировать будем веса
сигналов МА и MACD для получения максимального профита.
void FitnessFunction (int chromos)
{

double _MACD_Weight=0.0;
double _MA_Weight=0.0;
double sum=0.0;
int cnt=1;

while (cnt <=GeneCount)
{
_MACD_Weight=Colony [cnt] [chromos];
cnt++;
_MA_Weight=Colony [cnt] [chromos];
cnt++;

int handleInd;
double BufferInd [];
double BufferColorInd [];

handleInd=iCustom (NULL,0,«MAMACDExInd»,
Signal_ThresholdOpen,
Signal_ThresholdClose,
Signal_MACD_PeriodFast,
Signal_MACD_PeriodSlow,
Signal_MACD_PeriodSignal,
Signal_MACD_Applied,
```

```
    _MACD_Weight,
    Signal_MA_PeriodMA,
    Signal_MA_Shift,
    Signal_MA_Method,
    Signal_MA_Applied,
    _MA_Weight
);

ResetLastError ();

int size=5000;

if (CopyBuffer (handleInd,0,0,size, BufferInd) <0)
{
    // - - если копирование не удалось, сообщим код ошибки
    PrintFormat («Не удалось скопировать данные из индикатора 0, код ошибки %d»,
GetLastError ());
}

if (CopyBuffer (handleInd,1,0,size, BufferColorInd) <0)
{
    // - - если копирование не удалось, сообщим код ошибки
    PrintFormat («Не удалось скопировать данные из индикатора 1, код ошибки %d»,
GetLastError ());
}

ArraySetAsSeries (BufferInd, true);
ArraySetAsSeries (BufferColorInd, true);

// - - флаг покупки
bool flagBuy=false;
// - - флаг продажи
bool flagSell=false;
// - - покупка
double priceBuy=0;
double priceStopBuy=0;
double profitBuy=0;
double profitTotalBuy=0;
int countProfitBuyPlus=0;
int countProfitBuyMinus=0;
// - - продажа
double priceSell=0;
double priceStopSell=0;
double profitSell=0;
double profitTotalSell=0;
int countProfitSellPlus=0;
int countProfitSellMinus=0;
// - - спред
```

```
double sp=0.0002;

for (int i=0; i <size; i++) {

    if (BufferColorInd [i] ==2)
    {
        if (flagSell==true) {
            flagSell=false;
            priceStopSell=BufferInd [i];
            profitSell= (priceStopSell-priceSell-sp) *10000;
            if (profitSell> 0) {countProfitSellPlus++;} else {countProfitSellMinus++;}
            profitTotalSell=profitTotalSell+profitSell;
        }
        if (flagBuy==false) {
            flagBuy=true;
            priceBuy=BufferInd [i];
        }
    }

    if (BufferColorInd [i] ==1) {
        if (flagBuy==true) {
            flagBuy=false;
            priceStopBuy=BufferInd [i];
            profitBuy= (priceStopBuy-priceBuy-sp) *10000;
            if (profitBuy> 0) {countProfitBuyPlus++;} else {countProfitBuyMinus++;}
            profitTotalBuy=profitTotalBuy+profitBuy;
        }
        if (flagSell==false) {
            priceSell=BufferInd [i];
            flagSell=true;
        }
    }
}

//Print (« ProfitBuy», profitTotalBuy,» countProfitBuyPlus», countProfitBuyPlus,»
countProfitBuyMinus», countProfitBuyMinus);
//Print (« ProfitSell», profitTotalSell,» countProfitSellPlus», countProfitSellPlus,»
countProfitSellMinus», countProfitSellMinus);

sum = profitTotalBuy + profitTotalSell;
}
AmountStartsFF++;
Colony [0] [chromos] =sum;
}

void ServiceFunction ()
{

double _MACD_Weight=0.0;
double _MA_Weight=0.0;
```

```
int cnt=1;
```

```
while (cnt <=GeneCount)
{
    _MACD_Weight=Chromosome [cnt];
    cnt++;
    _MA_Weight=Chromosome [cnt];
    cnt++;
}
Print («Fitness func =», Chromosome [0],»\n»,
«Полученные значения аргументов:»,»\n»,
«_MACD_Weight =», Chromosome [1],»\n»,
«_MA_Weight =», Chromosome [2],»\n»
);

}
```

```
// -----
```

Напишем скрипт, который будет оптимизировать параметры индикатора с помощью генетического алгоритма.

```
#include <Fitness\MAMACDFitness.mqh>
```

```
#include <Fitness\UGAlib.mqh>
```

```
double ReplicationPortion_E = 100.0; //Доля Репликации.
```

```
double NMutationPortion_E = 10.0; //Доля Естественной мутации.
```

```
double ArtificialMutation_E = 10.0; //Доля Искусственной мутации.
```

```
double GenoMergingPortion_E = 20.0; //Доля Заимствования генов.
```

```
double CrossingOverPortion_E = 20.0; //Доля Кроссинговера.
```

```
// — —
```

```
double ReplicationOffset_E = 0.5; //Коэффициент смещения границ интервала
```

```
double NMutationProbability_E= 5.0; //Вероятность мутации каждого гена в %
```

```
// - - inputs for main signal
```

```
input int Signal_ThresholdOpen =20; // Signal threshold value to open [0...100]
```

```
input int Signal_ThresholdClose =20; // Signal threshold value to close [0...100]
```

```
input double Signal_PriceLevel =0.0; // Price level to execute a deal
```

```
input double Signal_StopLevel =50.0; // Stop Loss level (in points)
```

```
input double Signal_TakeLevel =50.0; // Take Profit level (in points)
```

```
input int Signal_Expiration =4; // Expiration of pending orders (in bars)
```

```
input int Signal_MA_PeriodMA =12; // Moving Average (12,0,...) Period of averaging
```

```
input int Signal_MA_Shift =0; // Moving Average (12,0,...) Time shift
```

```
input ENUM_MA_METHOD Signal_MA_Method =MODE_SMA; // Moving Average
(12,0,...) Method of averaging
```

```
input ENUM_APPLIED_PRICE Signal_MA_Applied =PRICE_CLOSE; // Moving
Average (12,0,...) Prices series
```

```
input double Signal_MA_Weight =1.0; // Moving Average (12,0,...) Weight [0...1.0]
```

```
input int Signal_MACD_PeriodFast =12; // MACD (12,24,9,PRICE_CLOSE) Period
of fast EMA
```

```
input int Signal_MACD_PeriodSlow =24; // MACD (12,24,9,PRICE_CLOSE) Period
of slow EMA
```

```

    input int Signal_MACD_PeriodSignal=9; // MACD (12,24,9,PRICE_CLOSE) Period
of averaging of difference
    input ENUM_APPLIED_PRICE Signal_MACD_Applied =PRICE_CLOSE; // MACD
(12,24,9,PRICE_CLOSE) Prices series
    input double Signal_MACD_Weight=1.0; // MACD (12,24,9,PRICE_CLOSE) Weight [0...
1.0]

//+-----+
//| Script program start function |
//+-----+
void OnStart ()
{
    // ---
    ChromosomeCount = 10; //Кол-во хромосом в колонии
    GeneCount = 2; //Кол-во генов
    Epoch = 50; //Кол-во эпох без улучшения
    // ---
    RangeMinimum = 0.0; //Минимум диапазона поиска
    RangeMaximum = 1.0; //Максимум диапазона поиска
    Precision = 0.1; //Требуемая точность
    OptimizeMethod = 2; //Оптим.:1-Min, другое-Max
    ArrayResize (Chromosome, GeneCount+1);
    ArrayInitialize (Chromosome,0);

    //Локальные переменные
    int time_start= (int) GetTickCount (),time_end=0;
    // -----

    //Запуск главной ф-ии UGA
    UGA
    (
        ReplicationPortion_E, //Доля Репликации.
        NMutationPortion_E, //Доля Естественной мутации.
        ArtificialMutation_E, //Доля Искусственной мутации.
        GenoMergingPortion_E, //Доля Заимствования генов.
        CrossingOverPortion_E, //Доля Кроссинговера.
        // ---
        ReplicationOffset_E, //Коэффициент смещения границ интервала
        NMutationProbability_E //Вероятность мутации каждого гена в %
    );
    // -----
    time_end= (int) GetTickCount ();
    // -----
    Print (time_end-time_start,» мс – Время исполнения»);
    // -----

}
//+-----+
Запустив скрипт, получим следующий результат:

```

```
Fitness func =402.2999999999981
```

```
Полученные значения аргументов:
```

```
_MACD_Weight =0.1
```

```
_MA_Weight =0.4
```

```
11498 мс – Время исполнения
```

Используемый индикатор основан на применении классов CиMA и CиMACD, имеющих проблемы с глубиной истории, потому параметр size в фитнес функции не может быть большим.

Перепишем индикатор, используя более низкоуровневый интерфейс.

```
#property indicator_chart_window
```

```
#include <Expert\ExpertInd.mqh>
```

```
#include <Expert\Signal\MySignal\SignalMACDInd.mqh>
```

```
#include <Expert\Signal\MySignal\SignalMAInd.mqh>
```

```
#property indicator_buffers 2
```

```
#property indicator_plots 1
```

```
#property indicator_type1 DRAW_COLOR_LINE
```

```
#property indicator_color1 clrBlack, clrRed, clrLawnGreen
```

```
#property indicator_style1 STYLE_SOLID
```

```
#property indicator_width1 2
```

```
double InBuffer [];
```

```
double ColorBuffer [];
```

```
int bars_calculated=0;
```

```
input int Signal_ThresholdOpen =20; // Signal threshold value to open [0...100]
```

```
input int Signal_ThresholdClose =20; // Signal threshold value to close [0...100]
```

```
input int Signal_MACD_PeriodFast =12; // MACD (12,24,9,PRICE_CLOSE) Period of fast EMA
```

```
input int Signal_MACD_PeriodSlow =24; // MACD (12,24,9,PRICE_CLOSE) Period of slow EMA
```

```
input int Signal_MACD_PeriodSignal =9; // MACD (12,24,9,PRICE_CLOSE) Period of averaging of difference
```

```
input ENUM_APPLIED_PRICE Signal_MACD_Applied =PRICE_CLOSE; // MACD (12,24,9,PRICE_CLOSE) Prices series
```

```
input double Signal_MACD_Weight =1.0; // MACD (12,24,9,PRICE_CLOSE) Weight [0...1.0]
```

```
input int Signal_MA_PeriodMA =12; // Moving Average (12,0,...) Period of averaging
```

```
input int Signal_MA_Shift =0; // Moving Average (12,0,...) Time shift
```

```
input ENUM_MA_METHOD Signal_MA_Method =MODE_SMA; // Moving Average (12,0,...) Method of averaging
```

```
input ENUM_APPLIED_PRICE Signal_MA_Applied =PRICE_CLOSE; // Moving Average (12,0,...) Prices series
```

```
input double Signal_MA_Weight =1.0; // Moving Average (12,0,...) Weight [0...1.0]
```

```
CExpertInd ExtExpert;
```

```
CSignalMAInd *filter0 = new CSignalMAInd;
```

```
CSignalMACDInd *filter1 = new CSignalMACDInd;

//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit ()
{

// - - Initializing expert
if(!ExtExpert.Init (Symbol (),Period (),true,100))
{
// - - failed
printf (__FUNCTION__+": error initializing expert»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// - - Creating signal
CExpertSignal *signal=new CExpertSignal;
if (signal==NULL)
{
// - - failed
printf (__FUNCTION__+": error creating signal»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// - -
ExtExpert.InitSignal (signal);

filter0.PeriodMA (Signal_MA_PeriodMA);
filter0.Shift (Signal_MA_Shift);
filter0.Method (Signal_MA_Method);
filter0.Applied (Signal_MA_Applied);

filter1.PeriodFast (Signal_MACD_PeriodFast);
filter1.PeriodSlow (Signal_MACD_PeriodSlow);
filter1.PeriodSignal (Signal_MACD_PeriodSignal);
filter1.Applied (Signal_MACD_Applied);

signal.AddFilter (filter0);
signal.AddFilter (filter1);
if(!ExtExpert.ValidationSettings ())
{
// - - failed
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// - - Tuning of all necessary indicators
if(!ExtExpert.InitIndicators ())
{
```

```
// - - failed
printf (__FUNCTION__+": error initializing indicators»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// - - ok
// - - indicator buffers mapping
SetIndexBuffer (0,InBuffer, INDICATOR_DATA);
SetIndexBuffer (1,ColorBuffer, INDICATOR_COLOR_INDEX);

ArraySetAsSeries (InBuffer, true);
ArraySetAsSeries (ColorBuffer, true);

// - —
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 [])
{
// - - количество копируемых значений из индикатора
int values_to_copy;
// - - узнаем количество рассчитанных значений в индикаторе
int calculated=MathMin(filter0.BarsCalculatedInd (), filter1.BarsCalculatedInd ());
if (calculated <=0)
{
PrintFormat («BarsCalculated () вернул %d, код ошибки %d», calculated, GetLastError ());
return (0);
}
// - - если это первый запуск вычислений нашего индикатора или изменилось количе-
ство значений в индикаторе
// - - или если необходимо рассчитать индикатор для двух или более баров (значит что-
то изменилось в истории)
if (prev_calculated==0 || calculated!=bars_calculated || rates_total> prev_calculated+1)
{
// - - если массив больше, чем значений в индикаторе на паре symbol/period, то копи-
руем не все
// - - в противном случае копировать будем меньше, чем размер индикаторных буферов
if (calculated> rates_total) values_to_copy=rates_total;
```

```
    else values_to_copy=calculated;
    }
    else
    {
        // - - значит наш индикатор рассчитывается не в первый раз и с момента последнего
        // вызова OnCalculate ()
        // - - для расчѐта добавилось не более одного бара
        values_to_copy= (rates_total-prev_calculated) +1;
    }

    bars_calculated=calculated;

    ArraySetAsSeries (open, true);
    ArraySetAsSeries (close, true);
    ArraySetAsSeries (low, true);
    ArraySetAsSeries (high, true);

    double _low [];
    ArrayCopy (_low, low);
    double _high [];
    ArrayCopy (_high, high);

    ArraySetAsSeries (_low, true);
    ArraySetAsSeries (_high, true);

    if (values_to_copy> 1)
    {

        ExtExpert.RefreshInd ();

        // - - флаг покупки
        bool flagBuy=false;
        // - флаг продажи
        bool flagSell=false;
        // - покупка
        double priceBuy=0;
        double priceStopBuy=0;
        double profitBuy=0;
        double profitTotalBuy=0;
        int countProfitBuyPlus=0;
        int countProfitBuyMinus=0;
        // - продажа
        double priceSell=0;
        double priceStopSell=0;
        double profitSell=0;
        double profitTotalSell=0;
        int countProfitSellPlus=0;
        int countProfitSellMinus=0;
        // - спред
```

```
double sp=0.0002;

for (int i=0; i <(values_to_copy-2); i++) {

    ColorBuffer [i] =0;
    InBuffer [i] =open [i];

    double result0=Signal_MA_Weight* (filter0.LongConditionInd (i, values_to_copy, close [i],
open [i], low[i])-filter0.ShortConditionInd (i, values_to_copy, close [i], open [i], high [i]));
    double result1=Signal_MACD_Weight* (filter1.LongConditionInd (i, values_to_copy,
_low, _high)-filter1.ShortConditionInd (i, values_to_copy, _low, _high));
    double result= (result0+result1) /2;

    if (result> =Signal_ThresholdOpen)
    {
        ColorBuffer [i] =2;
        if (flagSell==true) {
            flagSell=false;
            priceStopSell=InBuffer [i];
            profitSell= (priceStopSell-priceSell-sp) *10000;
            if (profitSell> 0) {countProfitSellPlus++;} else {countProfitSellMinus++;}
            profitTotalSell=profitTotalSell+profitSell;
        }
        if (flagBuy==false) {
            flagBuy=true;
            priceBuy=InBuffer [i];
        }
    }

    if (-result> =Signal_ThresholdOpen) {
        ColorBuffer [i] =1;
        if (flagBuy==true) {
            flagBuy=false;
            priceStopBuy=InBuffer [i];
            profitBuy= (priceStopBuy-priceBuy-sp) *10000;
            if (profitBuy> 0) {countProfitBuyPlus++;} else {countProfitBuyMinus++;}
            profitTotalBuy=profitTotalBuy+profitBuy;
        }
        if (flagSell==false) {
            priceSell=InBuffer [i];
            flagSell=true;
        }
    }

}

//Print (« ProfitBuy», profitTotalBuy,» countProfitBuyPlus», countProfitBuyPlus,»
countProfitBuyMinus», countProfitBuyMinus);
```

```
//Print (« ProfitSell», profitTotalSell,» countProfitSellPlus», countProfitSellPlus,»
countProfitSellMinus», countProfitSellMinus);

}
// -- return value of prev_calculated for next call
return (rates_total);
}
//+-----+
#include <Expert\Signal\SignalMA.mqh>
class CSignalMAInd: public CSignalMA
{
public:
virtual int BarsCalculatedInd ();
virtual int LongConditionInd (int ind, int amount, double close, double open, double low);
virtual int ShortConditionInd (int ind, int amount, double close, double open, double high);
};

//+-----+
//| Refresh indicators. |
//+-----+
int CSignalMAInd::BarsCalculatedInd () {
m_ma.Refresh ();
int bars = m_ma.BarsCalculated ();
return bars;
}
//+-----+
//| «Voting» that price will grow. |
//+-----+
int CSignalMAInd::LongConditionInd (int idx, int amount, double close, double open,
double low)
{
int handle=m_ma. Handle ();
double iMABuffer [];
if (CopyBuffer (handle,0,0,amount, iMABuffer) <0)
{
// -- если копирование не удалось, сообщим код ошибки
PrintFormat («Не удалось скопировать данные из индикатора iMA, код ошибки %d»,
GetLastError ());
// -- завершим с нулевым результатом – это означает, что индикатор будет считаться
нерассчитанным
return (-1);
}
ArraySetAsSeries (iMABuffer, true);

int result=0;

double DiffCloseMA = close – iMABuffer [idx];
double DiffOpenMA = open – iMABuffer [idx];
double DiffMA = iMABuffer [idx] – iMABuffer [idx+1];
```

```
double DiffLowMA = low - iMABuffer [idx];
// -- analyze positional relationship of the close price and the indicator at the first analyzed bar
if (DiffCloseMA <0.0)
{
    // -- the close price is below the indicator
    if (IS_PATTERN_USAGE (1) && DiffOpenMA> 0.0 && DiffMA> 0.0)
    {
        // -- the open price is above the indicator (i.e. there was an intersection), but the indicator
        is directed upwards
        result=m_pattern_1;
        // -- consider that this is an unformed «piercing» and suggest to enter the market at the
        current price
        m_base_price=0.0;
    }
}
else
{
    // -- the close price is above the indicator (the indicator has no objections to buying)
    if (IS_PATTERN_USAGE (0))
    result=m_pattern_0;
    // -- if the indicator is directed upwards
    if (DiffMA> 0.0)
    {
        if (DiffOpenMA <0.0)
        {
            // -- if the model 2 is used
            if (IS_PATTERN_USAGE (2))
            {
                // -- the open price is below the indicator (i.e. there was an intersection)
                result=m_pattern_2;
                // -- suggest to enter the market at the «roll back»
                m_base_price=m_symbol.NormalizePrice (iMABuffer [idx]);
            }
        }
    }
    else
    {
        // -- if the model 3 is used and the open price is above the indicator
        if (IS_PATTERN_USAGE (3) && DiffLowMA <0.0)
        {
            // -- the low price is below the indicator
            result=m_pattern_3;
            // -- consider that this is a formed «piercing» and suggest to enter the market at the current
            price
            m_base_price=0.0;
        }
    }
}
// -- return the result
```

```
    return (result);
}
//+-----+
//| «Voting» that price will fall. |
//+-----+
int CSignalMAInd::ShortConditionInd (int idx, int amount, double close, double open,
double high)
{
    int handle=m_ma. Handle ();
    double iMABuffer [];
    if (CopyBuffer (handle,0,0,amount, iMABuffer) <0)
    {
        // - - если копирование не удалось, сообщим код ошибки
        PrintFormat («Не удалось скопировать данные из индикатора iMA, код ошибки %d»,
GetLastError ());
        // - - завершим с нулевым результатом – это означает, что индикатор будет считаться
нерассчитанным
        return (-1);
    }
    ArraySetAsSeries (iMABuffer, true);

    int result=0;

    double DiffCloseMA = close – iMABuffer [idx];
    double DiffOpenMA = open – iMABuffer [idx];
    double DiffMA = iMABuffer [idx] – iMABuffer [idx+1];
    double DiffHighMA = high – iMABuffer [idx];

    // - - analyze positional relationship of the close price and the indicator at the first analyzed bar
    if (DiffCloseMA > 0.0)
    {
        // - - the close price is above the indicator
        if (IS_PATTERN_USAGE (1) && DiffOpenMA <0.0 && DiffMA <0.0)
        {
            // - - the open price is below the indicator (i.e. there was an intersection), but the indicator
is directed downwards
            result=m_pattern_1;
            // - - consider that this is an unformed «piercing» and suggest to enter the market at the
current price
            m_base_price=0.0;
        }
    }
    else
    {
        // - - the close price is below the indicator (the indicator has no objections to buying)
        if (IS_PATTERN_USAGE (0))
            result=m_pattern_0;
        // - - the indicator is directed downwards
        if (DiffMA <0.0)
```

```

    {
    if (DiffOpenMA> 0.0)
    {
    // - - if the model 2 is used
    if (IS_PATTERN_USAGE (2))
    {
    // - - the open price is above the indicator (i.e. there was an intersection)
    result=m_pattern_2;
    // - - suggest to enter the market at the «roll back»
    m_base_price=m_symbol.NormalizePrice (iMABuffer [idx]);
    }
    }
    else
    {
    // - - if the model 3 is used and the open price is below the indicator
    if (IS_PATTERN_USAGE (3) && DiffHighMA> 0.0)
    {
    // - - the high price is above the indicator
    result=m_pattern_3;
    // - - consider that this is a formed «piercing» and suggest to enter the market at the current
price
    m_base_price=0.0;
    }
    }
    }
    // - - return the result
    return (result);
    }
    //+-----+
    #include <Expert\Signal\SignalMACD.mqh>

    class CSignalMACDInd: public CSignalMACD
    {
    public:
    virtual int BarsCalculatedInd ();
    virtual int LongConditionInd (int ind, int amount, double &low [], double &high []);
    virtual int ShortConditionInd (int ind, int amount, double &low [], double &high []);
    protected:
    int StateMain (int ind, double &Main []);
    bool ExtState (int ind, double &Main [], double &low [], double &high []);
    };
    //+-----+
    //| Refresh indicators. |
    //+-----+
    int CSignalMACDInd:: BarsCalculatedInd () {
    m_MACD.Refresh ();
    int bars = m_MACD.BarsCalculated ();
    return bars;
    }

```

```
    }

    //+-----+
    //| «Voting» that price will grow. |
    //+-----+
    int CSignalMACDInd::LongConditionInd (int idx, int amount, double &low [], double
&high [])
    {

        int handle=m_MACD. Handle ();
        double MACDBuffer [];
        double SignalBuffer [];
        if (CopyBuffer (handle,0,0,amount, MACDBuffer) <0)
        {
            // - если копирование не удалось, сообщим код ошибки
            PrintFormat («Не удалось скопировать данные из индикатора iMA, код ошибки %d»,
GetLastError ());
            // - завершим с нулевым результатом – это означает, что индикатор будет считаться
нерассчитанным
            return (-1);
        }
        if (CopyBuffer (handle,1,0,amount, SignalBuffer) <0)
        {
            // - если копирование не удалось, сообщим код ошибки
            PrintFormat («Не удалось скопировать данные из индикатора iMA, код ошибки %d»,
GetLastError ());
            // - завершим с нулевым результатом – это означает, что индикатор будет считаться
нерассчитанным
            return (-1);
        }
        ArraySetAsSeries (MACDBuffer, true);
        ArraySetAsSeries (SignalBuffer, true);
        int result=0;

        double DiffMain = MACDBuffer [idx] -MACDBuffer [idx+1];
        double DiffMain_1 = MACDBuffer [idx+1] -MACDBuffer [idx+2];
        double State = MACDBuffer [idx] – SignalBuffer [idx];
        double State_1 = MACDBuffer [idx+1] – SignalBuffer [idx+1];

        // - - check direction of the main line
        if (DiffMain> 0.0)
        {
            // - - the main line is directed upwards, and it confirms the possibility of price growth
            if (IS_PATTERN_USAGE (0))
                result=m_pattern_0; // «confirming» signal number 0
            // - - if the model 1 is used, look for a reverse of the main line
            if (IS_PATTERN_USAGE (1) && DiffMain_1 <0.0)
                result=m_pattern_1; // signal number 1
            // - - if the model 2 is used, look for an intersection of the main and signal line
```

```
if (IS_PATTERN_USAGE (2) && State> 0.0 && State_1 <0.0)
result=m_pattern_2; // signal number 2
// - - if the model 3 is used, look for an intersection of the main line and the zero level
if (IS_PATTERN_USAGE (3) && MACDBuffer [idx]> 0.0 && MACDBuffer [idx+1] <0.0)
result=m_pattern_3; // signal number 3
// - - if the models 4 or 5 are used and the main line turned upwards below the zero level,
look for divergences
if ((IS_PATTERN_USAGE (4) || IS_PATTERN_USAGE (5)) && MACDBuffer [idx] <0.0)
{
// - - perform the extended analysis of the oscillator state
ExtState (idx, MACDBuffer, low, high);
// - - if the model 4 is used, look for the «divergence» signal
if (IS_PATTERN_USAGE (4) && CompareMaps (1,1)) // 0000 0001b
result=m_pattern_4; // signal number 4
// - - if the model 5 is used, look for the «double divergence» signal
if (IS_PATTERN_USAGE (5) && CompareMaps (0x11,2)) // 0001 0001b
return (m_pattern_5); // signal number 5
}
}
// - - return the result
return (result);
}
//+-----+
//| «Voting» that price will fall. |
//+-----+
int CSignalMACDInd::ShortConditionInd (int idx, int amount, double &low [], double
&high [])
{
int handle=m_MACD. Handle ();
double MACDBuffer [];
double SignalBuffer [];
if (CopyBuffer (handle,0,0,amount, MACDBuffer) <0)
{
// - - если копирование не удалось, сообщим код ошибки
PrintFormat («Не удалось скопировать данные из индикатора iMA, код ошибки %d»,
GetLastError ());
// - - завершим с нулевым результатом – это означает, что индикатор будет считаться
нерассчитанным
return (-1);
}
if (CopyBuffer (handle,1,0,amount, SignalBuffer) <0)
{
// - - если копирование не удалось, сообщим код ошибки
PrintFormat («Не удалось скопировать данные из индикатора iMA, код ошибки %d»,
GetLastError ());
// - - завершим с нулевым результатом – это означает, что индикатор будет считаться
нерассчитанным
return (-1);
}
```

```
ArraySetAsSeries (MACDBuffer, true);
ArraySetAsSeries (SignalBuffer, true);
int result=0;

double DiffMain = MACDBuffer [idx] -MACDBuffer [idx+1];
double DiffMain_1 = MACDBuffer [idx+1] -MACDBuffer [idx+2];
double State = MACDBuffer [idx] – SignalBuffer [idx];
double State_1 = MACDBuffer [idx+1] – SignalBuffer [idx+1];

// - - check direction of the main line
if (DiffMain <0.0)
{
// - - main line is directed downwards, confirming a possibility of falling of price
if (IS_PATTERN_USAGE (0))
result=m_pattern_0; // «confirming» signal number 0
// - - if the model 1 is used, look for a reverse of the main line
if (IS_PATTERN_USAGE (1) && DiffMain_1> 0.0)
result=m_pattern_1; // signal number 1
// - - if the model 2 is used, look for an intersection of the main and signal line
if (IS_PATTERN_USAGE (2) && State <0.0 && State_1> 0.0)
result=m_pattern_2; // signal number 2
// - - if the model 3 is used, look for an intersection of the main line and the zero level
if (IS_PATTERN_USAGE (3) && MACDBuffer [idx] <0.0 && MACDBuffer [idx+1]> 0.0)
result=m_pattern_3; // signal number 3
// - - if the models 4 or 5 are used and the main line turned downwards above the zero level,
look for divergences
if ((IS_PATTERN_USAGE (4) || IS_PATTERN_USAGE (5)) && MACDBuffer [idx]> 0.0)
{
// - - perform the extended analysis of the oscillator state
ExtState (idx, MACDBuffer, low, high);
// - - if the model 4 is used, look for the «divergence» signal
if (IS_PATTERN_USAGE (4) && CompareMaps (1,1)) // 0000 0001b
result=m_pattern_4; // signal number 4
// - - if the model 5 is used, look for the «double divergence» signal
if (IS_PATTERN_USAGE (5) && CompareMaps (0x11,2)) // 0001 0001b
return (m_pattern_5); // signal number 5
}
}
// - - return the result
return (result);
}
//+-----+
//+-----+
//| Check of the oscillator state. |
//+-----+
int CSignalMACDInd::StateMain (int ind, double &Main [])
{
int res=0;
double var;
```

```

// — —
for (int i=ind;;i++)
{
if (Main [i+1] ==EMPTY_VALUE)
break;
var= (Main [i] -Main [i+1]);
if (res> 0)
{
if (var <0)
break;
res++;
continue;
}
if (res <0)
{
if (var> 0)
break;
res — ;
continue;
}
if (var> 0)
res++;
if (var <0)
res — ;
}
// — —
return (res);
}
//+ — — — — — +
//| Extended check of the oscillator state consists |
//| in forming a bit-map according to certain rules, |
//| which shows ratios of extremums of the oscillator and price. |
//+ — — — — — +
bool CSignalMACDInd::ExtState (int ind, double &Main [], double &low [], double
&high [])
{
// — - operation of this method results in a bit-map of extremums
// — - practically, the bit-map of extremums is an «array» of 4-bit fields
// — - each «element of the array» definitely describes the ratio
// — - of current extremums of the oscillator and the price with previous ones
// — - purpose of bits of an element of the analyzed bit-map
// — - bit 3 – not used (always 0)
// — - bit 2 – is equal to 1 if the current extremum of the oscillator is «more extreme» than
the previous one
// — - (a higher peak or a deeper valley), otherwise – 0
// — - bit 1 – not used (always 0)
// — - bit 0 – is equal to 1 if the current extremum of price is «more extreme» than the
previous one
// — - (a higher peak or a deeper valley), otherwise – 0

```

```
// - - in addition to them, the following is formed:
// - - array of values of extremums of the oscillator,
// - - array of values of price extremums and
// - - array of «distances» between extremums of the oscillator (in bars)
// - - it should be noted that when using the results of the extended check of state,
// - - you should consider, which extremum of the oscillator (peak or valley)
// - - is the «reference point» (i.e. was detected first during the analysis)
// - - if a peak is detected first then even elements of all arrays
// - - will contain information about peaks, and odd elements will contain information about
valleys
// - - if a valley is detected first, then respectively in reverse
int pos=ind, off, index;
uint map; // intermediate bit-map for one extremum
// - —
m_extr_map=0;
for (int i=0;i <10;i++)
{
off=StateMain (pos, Main);
if (off> 0)
{
// - - minimum of the oscillator is detected
pos+=off;
m_extr_pos [i] =pos;
m_extr_osc [i] =Main [pos];
if (i> 1)
{
index = ArrayMinimum (low, pos-2,5);
m_extr_pr [i] =low [index];
// - - form the intermediate bit-map
map=0;
if (m_extr_pr [i-2] <m_extr_pr [i])
map+=1; // set bit 0
if (m_extr_osc [i-2] <m_extr_osc [i])
map+=4; // set bit 2
// - - add the result
m_extr_map+=map <<(4* (i-2));
}
else
index = ArrayMinimum (low, pos-1,4);
m_extr_pr [i] =low [index];
}
else
{
// - - maximum of the oscillator is detected
pos-=off;
m_extr_pos [i] =pos;
m_extr_osc [i] =Main [pos];
if (i> 1)
{
```

```

index = ArrayMaximum (high, pos-2,5);
m_extr_pr [i] =high [index];
// - - form the intermediate bit-map
map=0;
if (m_extr_pr [i-2]> m_extr_pr [i])
map+=1; // set bit 0
if (m_extr_osc [i-2]> m_extr_osc [i])
map+=4; // set bit 2
// - - add the result
m_extr_map+=map <<(4* (i-2));
}
else
index = ArrayMaximum (high, pos-1,4);
m_extr_pr [i] =high [index];
}
}
// - -
return (true);
}

```

Включим этот индикатор в фитнес функцию.

```

handleInd=iCustom (NULL,0,«MAMACDIndicator»,
Signal_ThresholdOpen,
Signal_ThresholdClose,
Signal_MACD_PeriodFast,
Signal_MACD_PeriodSlow,
Signal_MACD_PeriodSignal,
Signal_MACD_Applied,
_MACD_Weight,
Signal_MA_PeriodMA,
Signal_MA_Shift,
Signal_MA_Method,
Signal_MA_Applied,
_MA_Weight
);

```

И запустим скрипт.

Глубина истории увеличится, но при этом существенно увеличится и время оптимизации – на порядок, при том же самом результате на 1000 барах. Поэтому для самооптимизации советника будем использовать первый вариант индикатора с глубиной истории в 1000 бар.

С помощью мастера MQL5 Wizard сгенерируем код советника на основе сигналов MA и MACD и добавим в него самооптимизацию параметров Signal_MA_Weight и Signal_MACD_Weight с использованием приведенной выше фитнес функции.

```

//+-----+
//| Include |
//+-----+
#include <Expert\Expert.mqh>
// - - available signals
#include <Expert\Signal\SignalMA.mqh>
#include <Expert\Signal\SignalMACD.mqh>
// - - available trailing

```

```
#include <Expert\Trailing\TrailingNone.mqh>
// -- available money management
#include <Expert\Money\MoneyFixedLot.mqh>

#include <Fitness\MAMACDFitness.mqh>
#include <Fitness\UGAlib.mqh>
//+-----+
//| Inputs |
//+-----+
// -- inputs for expert
input string Expert_Title =«MAMACDExpert»; // Document name
ulong Expert_MagicNumber =4322; //
bool Expert_EveryTick =false; //
// -- inputs for main signal
input int Signal_ThresholdOpen =20; // Signal threshold value to open [0...100]
input int Signal_ThresholdClose =20; // Signal threshold value to close [0...100]
input double Signal_PriceLevel =0.0; // Price level to execute a deal
input double Signal_StopLevel =50.0; // Stop Loss level (in points)
input double Signal_TakeLevel =50.0; // Take Profit level (in points)
input int Signal_Expiration =4; // Expiration of pending orders (in bars)
input int Signal_MA_PeriodMA =12; // Moving Average (12,0,...) Period of averaging
input int Signal_MA_Shift =0; // Moving Average (12,0,...) Time shift
input ENUM_MA_METHOD Signal_MA_Method =MODE_SMA; // Moving Average
(12,0,...) Method of averaging
input ENUM_APPLIED_PRICE Signal_MA_Applied =PRICE_CLOSE; // Moving
Average (12,0,...) Prices series
input double Signal_MA_Weight =1.0; // Moving Average (12,0,...) Weight [0...1.0]
input int Signal_MACD_PeriodFast =12; // MACD (12,24,9,PRICE_CLOSE) Period
of fast EMA
input int Signal_MACD_PeriodSlow =24; // MACD (12,24,9,PRICE_CLOSE) Period
of slow EMA
input int Signal_MACD_PeriodSignal=9; // MACD (12,24,9,PRICE_CLOSE) Period
of averaging of difference
input ENUM_APPLIED_PRICE Signal_MACD_Applied =PRICE_CLOSE; // MACD
(12,24,9,PRICE_CLOSE) Prices series
input double Signal_MACD_Weight =1.0; // MACD (12,24,9,PRICE_CLOSE) Weight [0...
1.0]
// -- inputs for money
input double Money_FixLot_Percent =10.0; // Percent
input double Money_FixLot_Lots =1.0; // Fixed volume
//+-----+
//| Global expert object |
//+-----+
CExpert ExtExpert;
CSignalMA *filter0=new CSignalMA;
CSignalMACD *filter1=new CSignalMACD;

double ReplicationPortion_E = 100.0; //Доля Репликации.
double NMutationPortion_E = 10.0; //Доля Естественной мутации.
```

```
double ArtificialMutation_E = 10.0; //Доля Искусственной мутации.
double GenoMergingPortion_E = 20.0; //Доля Заимствования генов.
double CrossingOverPortion_E = 20.0; //Доля Кроссинговера.
// — —

double ReplicationOffset_E = 0.5; //Коэффициент смещения границ интервала
double NMutationProbability_E= 5.0; //Вероятность мутации каждого гена в %

double balance;

//+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
//| Initialization function of the expert |
//+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +

int OnInit ()
{
// -- Initializing expert
if(!ExtExpert.Init (Symbol (),Period (),Expert_EveryTick, Expert_MagicNumber))
{
// -- failed
printf (__FUNCTION__+": error initializing expert»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- Creating signal
CExpertSignal *signal=new CExpertSignal;
if (signal==NULL)
{
// -- failed
printf (__FUNCTION__+": error creating signal»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// — —
ExtExpert.InitSignal (signal);
signal.ThresholdOpen (Signal_ThresholdOpen);
signal.ThresholdClose (Signal_ThresholdClose);
signal.PriceLevel (Signal_PriceLevel);
signal.StopLevel (Signal_StopLevel);
signal.TakeLevel (Signal_TakeLevel);
signal.Expiration (Signal_Expiration);
// -- Creating filter CSignalMA

if (filter0==NULL)
{
// -- failed
printf (__FUNCTION__+": error creating filter0»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
signal.AddFilter (filter0);
```

```
// -- Set filter parameters
filter0.PeriodMA (Signal_MA_PeriodMA);
filter0.Shift (Signal_MA_Shift);
filter0.Method (Signal_MA_Method);
filter0.Applied (Signal_MA_Applied);
filter0.Weight (Signal_MA_Weight);
// -- Creating filter CSignalMACD

if (filter1==NULL)
{
// -- failed
printf (__FUNCTION__+": error creating filter1»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
signal.AddFilter (filter1);
// -- Set filter parameters
filter1.PeriodFast (Signal_MACD_PeriodFast);
filter1.PeriodSlow (Signal_MACD_PeriodSlow);
filter1.PeriodSignal (Signal_MACD_PeriodSignal);
filter1.Applied (Signal_MACD_Applied);
filter1.Weight (Signal_MACD_Weight);
// -- Creation of trailing object
CTrailingNone *trailing=new CTrailingNone;
if (trailing==NULL)
{
// -- failed
printf (__FUNCTION__+": error creating trailing»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- Add trailing to expert (will be deleted automatically)
if(!ExtExpert.InitTrailing (trailing))
{
// -- failed
printf (__FUNCTION__+": error initializing trailing»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// -- Set trailing parameters
// -- Creation of money object
CMoneyFixedLot *money=new CMoneyFixedLot;
if (money==NULL)
{
// -- failed
printf (__FUNCTION__+": error creating money»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
```

```
// - - Add money to expert (will be deleted automatically))
if(!ExtExpert.InitMoney (money))
{
// - - failed
printf (__FUNCTION__+": error initializing money»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// - - Set money parameters
money.Percent (Money_FixLot_Percent);
money.Lots (Money_FixLot_Lots);
// - - Check all trading objects parameters
if(!ExtExpert.ValidationSettings ())
{
// - - failed
ExtExpert.Deinit ();
return (INIT_FAILED);
}
// - - Tuning of all necessary indicators
if(!ExtExpert.InitIndicators ())
{
// - - failed
printf (__FUNCTION__+": error initializing indicators»);
ExtExpert.Deinit ();
return (INIT_FAILED);
}

ChromosomeCount = 100; //Кол-во хромосом в колонии
GeneCount = 2; //Кол-во генов
Epoch = 50; //Кол-во эпох без улучшения
// - —
RangeMinimum = 0.0; //Минимум диапазона поиска
RangeMaximum = 1.0; //Максимум диапазона поиска
Precision = 0.1; //Требуемая точность
OptimizeMethod = 2; //Оптим.:1-Min, другое-Max
ArrayResize (Chromosome, GeneCount+1);
ArrayInitialize (Chromosome,0);

balance=AccountInfoDouble (ACCOUNT_BALANCE);

// - - ok
return (INIT_SUCCEEDED);
}
//+ - - - - - +
//| Deinitialization function of the expert |
//+ - - - - - +
void OnDeinit (const int reason)
{
ExtExpert.Deinit ();
```

```

    }
    //+-----+
    //| «Tick» event handler function |
    //+-----+
    void OnTick ()
    {
        if (AccountInfoDouble (ACCOUNT_BALANCE)> balance) balance=AccountInfoDouble
(Account_BALANCE);
        double bd = ((balance-AccountInfoDouble (ACCOUNT_BALANCE)) /balance) *100;

        if (bd> 10) {

            UGA
            (
                ReplicationPortion_E, //Доля Репликации.
                NMutationPortion_E, //Доля Естественной мутации.
                ArtificialMutation_E, //Доля Искусственной мутации.
                GenoMergingPortion_E, //Доля Заимствования генов.
                CrossingOverPortion_E, //Доля Кроссинговера.
                // — —
                ReplicationOffset_E, //Коэффициент смещения границ интервала
                NMutationProbability_E //Вероятность мутации каждого гена в %
            );

            double _MACD_Weight=0.0;
            double _MA_Weight=0.0;
            int cnt=1;

            while (cnt <=GeneCount)
            {
                _MACD_Weight=Chromosome [cnt];
                cnt++;
                _MA_Weight=Chromosome [cnt];
                cnt++;
            }
            filter0.Weight (_MA_Weight);
            filter1.Weight (_MACD_Weight);

            balance=AccountInfoDouble (ACCOUNT_BALANCE);
        }
        ExtExpert. OnTick ();
    }
    //+-----+
    //| «Trade» event handler function |
    //+-----+
    void OnTrade ()
    {
        ExtExpert. OnTrade ();
    }

```

```
//+-----+
//| «Timer» event handler function |
//+-----+
void OnTimer ()
{
    ExtExpert. OnTimer ();
}
//+-----+
```

Здесь в функции OnTick при превышении порога просадки баланса вызывается функция UGA генетического алгоритма, которая оптимизирует веса сигналов.

В фитнес функцию перед копированием буферов индикатора добавим вызов Sleep (1000); для того, чтобы индикатор успел рассчитаться.

При тестировании советника на паре EURUSD на периоде H1 (2016.01.01 – 2016.06.30) без самооптимизации получаем следующий результат:

Чистая прибыль: 26.00
Абсолютная просадка по балансу: 4 431.00
Абсолютная просадка по средствам: 4 550.00
Общая прибыль: 79 532.00
Максимальная просадка по балансу: 5 862.00 (50.08%)
Максимальная просадка по средствам: 5 956.00 (50.57%)
Общий убыток: -79 506.00
Относительная просадка по балансу: 50.08% (5 862.00)
Относительная просадка по средствам: 50.57% (5 956.00)
Прибыльность: 1.00
Матожидание выигрыша: 0.04
С включенной самооптимизацией советника при тестировании получаем результат:
Чистая прибыль: 384.00
Абсолютная просадка по балансу: 504.00
Абсолютная просадка по средствам: 612.00
Общая прибыль: 4 004.00
Максимальная просадка по балансу: 1 276.00 (10.94%)
Максимальная просадка по средствам: 1 341.00 (11.44%)
Общий убыток: -3 620.00
Относительная просадка по балансу: 10.94% (1 276.00)
Относительная просадка по средствам: 12.43% (1 333.00)
Прибыльность: 1.11
Матожидание выигрыша: 24.00