MQL4-программирование: торговый робот за один вечер

Евгений Ждан

Евгений Ждан

MQL4-программирование: торговый робот за один вечер

http://www.litres.ru/pages/biblio_book/?art=69176782
SelfPub; 2023

Аннотация

Чтобы освободиться от рутины и сберечь нервы, каждый трейдер задумывается об автоматизации своей торговли. Эта книга поможет освоить mql4-программирование любому человеку с любым уровнем образования. Книга написана простым языком без сложной терминологии. На протяжении всей книги автор и читатель вместе разрабатывают торговый советник для платформы MetaTrader4

Евгений Ждан
MQL4-программирование: торговый робот за один вечер

ВВЕДЕНИЕ

Рынок FOREX, как и биржа ценных бумаг привлекает все новых и новых людей. Это и понятно – делать деньги из воздуха хочется всем. Но, не все так просто.

Данные о том, что именно **зарабатывающих** трейдеров не более 5% встречаются повсеместно. Деньги остальных 95% делятся между первыми 5% и брокерами/диллинговыми центрами.

Чтобы попасть в число успешных трейдеров необходимо иметь четкую стратегию работы и железные нервы. В принципе, прибыльных торговых стратегий и систем существует не мало. Основная проблема работы заключается в психологии трейдера. Как правило, почти все трейдеры стартуют стабильно успешно. Немногим далее – постепенный или внезапный крах.

Дело в том, что, начиная торговать, трейдер выполняет условия своей торговой стратегии. После череды успешных

сделок, последний расслабляется, начинает думать, что поймал удачу за бороду и жизнь обеспечена. Появляется чрезмерная уверенность в своих действиях и трейдер начинает отклоняться от торговой стратегии. Открываются сделки не по системе, а по «наитию». Пребывая в эйфории от череды успешно закрытых ордеров, трейдер увеличивает торговый лот. И уже скоро этот человек отправляется в число тех теряющих деньги 95% трейдеров.

Решить проблему психологической стороны торговли может ее автоматизация – использование **торгового эксперта** (советника, торгового робота), который будет работать на счете трейдера без вмешательства человека – хозяина счета.

Торговый робот лишен эмоций и способен монотонно выполнять свой алгоритм с приходом каждого нового ценового значения. Конечно, иногда трейдеру придется запрещать ему работу, например, во время крайне-важных финансово-политических новостей, когда волатильность возрастает в разы. Например, такими событиями в недавнем прошлом являлись Britain Exit – «Брекзит» – кампания сторонников выхода Великобритании из ЕС, выборы президентов США и Франции, авария на АЭС Фукусима-1, спровоцировавшая обвал японской национальной валюты и т. п. Надеюсь, мысль вам понятна.

В этой книге мы научимся делать торговых роботов для для самого распространенного и самого удобного торгового терминала MetaTrader4 от компании MetaQuotes. Если

быть точнее, в этой книге мы по шагам создадим советника (Expert Advisor), полностью готового «к употреблению». Естественно, прибыльность конечного продукта я не обещаю, нам важно другое – научиться их делать.

После изучения этой книги Вы сможете воплощать свои самые смелые торговые идеи самостоятельно, не прибегая к услугам mql-программистов. Также, вы сможете и сами зарабатывать, программируя советники на заказ.

Пожалуй, уже к середине изучения данной книги вы будете отклоняться от нее и вносить свои коррективы в создаваемый нами советник. Так и должно быть. Поехали.

НЕМНОГО ТЕОРИИ

Типы данных

Торговый эксперт оперирует данными. Он работает с поступающими ценами, ценовыми значениями индикаторов, ведет подсчеты открытых ордеров, что-то печатает в Журнал торгового терминала.

В mql4 существуют следующие типы данных:

Основные типы данных:

целые числа (char, short, int, long, uchar, ushort, uint, ulong)

логические (bool)

литералы (ushort)

строки (string)

числа с плавающей точкой (double, float)

цвет (color)

дата и время (datetime)
перечисления (enum)
Сложные типы данных:
структуры;
классы.

На первых порах вам не понадобится и 70% из вышеперечисленного. Рассмотрим только то, что нам будет нужно в рамках разработки нашего торгового эксперта.

Тип int – целые числа, т. е. 1, 2, 5,100, 1425…

Тип double – числа с дробной частью (с запятой): 1,0254, 0,0547....

Тип bool – имеет только 2 значения – true (правда) и false (ложь).

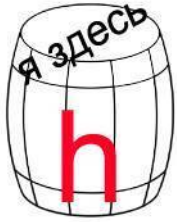
Тип string – строковые значения, т. е. слова: “слово”, “предложение из четырех слов”…

Переменные

Переменные – это буквенные символы, содержащие в себе значения какого-либо типа. Переменные – это бочонки, в которых что-то лежит.

int x = 100 означает, что в бочонке
по имени x лежит целое число 100;

double z = 0.03254 означает, что в бочонке
по имени z лежит число 0,03254



string h = “я здесь” означает, что в бочонке
лежит эта строка - “я здесь”

С типом
bool
все также, например переменная
bool
b
=

true
, означает, что бочонок с именем
b
содержит в себе
true
.

Перед тем как создавать переменную для последующей работы с ней, нужно обязательно объявить ее тип, чтобы компилятор редактора MetaEditor (в нем мы будем создавать нашего робота) знал, что в этой переменной будет храниться. Названия переменных не могут начинаться с цифры.

Объявлять переменную нужно только 1 раз. Позднее мы поговорим о том, где их можно объявлять и как это влияет на последующую работу.

Условные операторы if-else

Условные операторы if-else применяются всегда и везде. If – означает “если”, else – “если нет, то”.

Например:

if(x < y) // Если содержимое бочонка x меньше содержимого бочонка y

{

Что-то делаем, например, открываем ордер. Или закрываем другой ордер, да все что угодно!

}else // А если x не меньше y, делаем то, что ниже, в фигурных скобках

```
{
Делаем что-то здесь.
}
```

использование оператора else не обязательно, все зависит от конкретной задачи.

Два слеша (косые черты) - //, то что после них в коде советника означают **комментарии.** При компиляции вашего советника (превращения вашего кода в машинный код, понятный компьютеру), комментарии игнорируются. Комментарии желательно писать для себя, чтобы не забыть что куда и зачем сделано.

Блоки комментариев делаются так:

```
/* это
блок
комментария */
```

Все, что между символами **/* и */** также компилятором игнорируется.

Циклы

В mql4 существуют циклы for и while. Чаще используется for, но, нередко и while.

```
for (int i=0; i <100; i++)
{
    что-то считаем 100 раз.
}
```

int i = 0 – объявляем переменную, которая будет работать в пределах данного цикла; i < 100 – цикл прокрутится 100

раз, от 0 до 99; i++ (инкремент) означает, что при каждой прокрутке (итерации) цикла, переменная i будет увеличена на единицу.

bool x = false; //присваиваем переменной x типа bool значение false

while(x==false) //пока x равен false. Два символа равно "==" означают сравнение

{

/*

здесь будут выполняться какие-то условия.

Как только x станет true, цикл прекратится.

*/

//например

x = true;//после первого же прохода делаем x равным true

//и цикл прекращается

}

В процессе написания советника мы будем использовать оба этих цикла, и вы без труда с ними разберетесь.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Опишем, что и когда должен делать наш будущий советник:

Торговые сигналы будут формировать два стандартных индикатора Envelopes
и
ZigZag

. Эти индикаторы встроены в
MetaTrader
4 и дополнительно скачивать их не нужно. Я выбрал два именно этих индикатора, т. к. их значения вызываются разными способами. Для
Envelopes
– с помощью стандартной функции iEnvelopes, а
ZigZag
вызывается функцией
iCustom
– ее вам необходимо изучить (хотя, это громко сказано), чтобы в дальнейшем вы умели вызывать данные почти
любых не стандартных
индикаторов для
MetaTrader
4.
Составим краткое техническое задание:
1) Если верхний пик индикатора
ZigZag
(далее -
ZZ
) сформировался выше верхней линии индикатора
Envelopes
(с параметром
Shift
= 10, остальные – стандартные), выставляем ордер на про-

дажу фиксированным лотом, определенным в настройках советника.

2) Если нижний
пик
ZZ
сформировался ниже нижней
Envelopes
– сигнал на покупку (т. е. наоборот от
buy
–сигнала).

3) Путем модификации (почему модификации, а не сразу при установке ордера – позже, когда будем писать этот код) советник должен устанавливать у ордеров Стоп-Лосс и Тейк-Профит.

4) Добавить возможность закрывать ордера при касании ценой противоположной линии Envelopes. Эту функцию можно выключать в настройках.

Если вы читаете эту книгу, я надеюсь, на вашем компьютере уже установлен торговый терминал MetaTrader4 и вы умеете открывать демо-счет. Если нет, нужно этот терминал установить, предварительно зарегистрировавшись у любого брокера, поддерживающего работу MetaTrader4.

А теперь, **переведите свой терминал на английский язык!** Если уж вы решили заняться программированием, привыкайте к английскому, без этого никуда! Сам редактор кода MetaEditor лучше оставить на русском, т. к. при пере-

воде его на английский язык, Справка (F1) получается тоже на английском языке. Не всем это удобно.

ПОЛУЧАЕМ ДАННЫЕ ИНДИКАТОРОВ

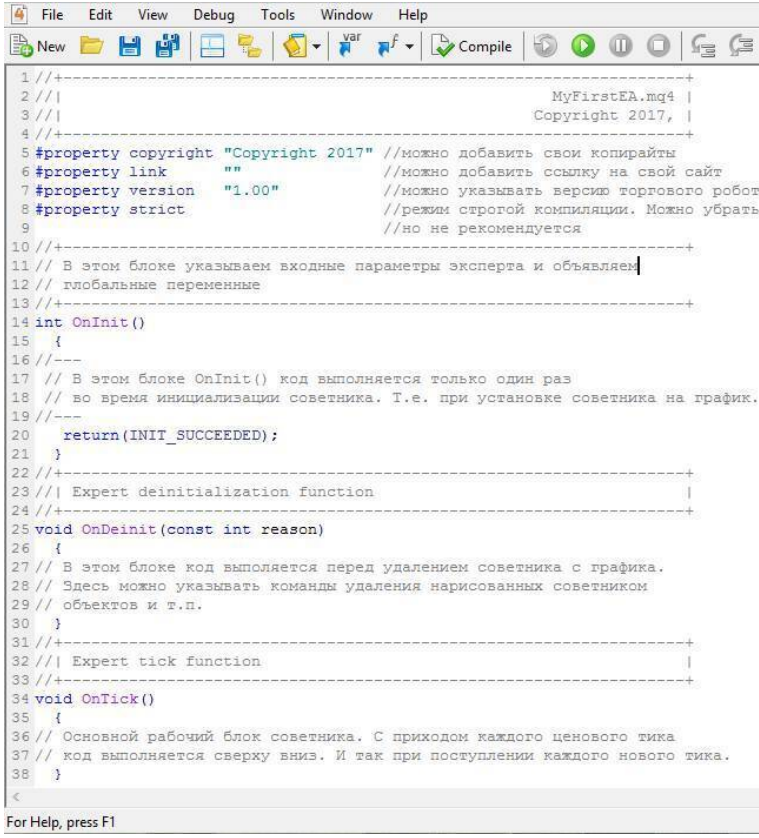
Открываем свой MetaTrader4 и нажимаем кнопку F4 на клавиатуре, или левой кнопкой мыши здесь:

В открывшемся редакторе кода нажимаем
New
(Создать), затем
Expert
Advisor
(
template
), потом Далее, в поле
Name
после
Experts
\ дописываем
MyFirstEA
– это будет названием вашего первого советника. Полу-

чится Experts\MyFirstEA. Поля
Autor
,
link
нам в этом учебном советнике не важны. Нажимаем
кнопку
Далее
.
Появится
окошко
Event Handles of the Expert Advisor.

Здесь ничего отмечать не нужно и просто нажмем Далее. В появившемся окошке
Tester
event
handless
of
the
Expert
Advisor
снова ничего не выбираем и жмем Готово. Получаем рабочую область, в которой скоро родится наш торговый робот.

На изображении в комментариях указано, какие блоки за что отвечают.



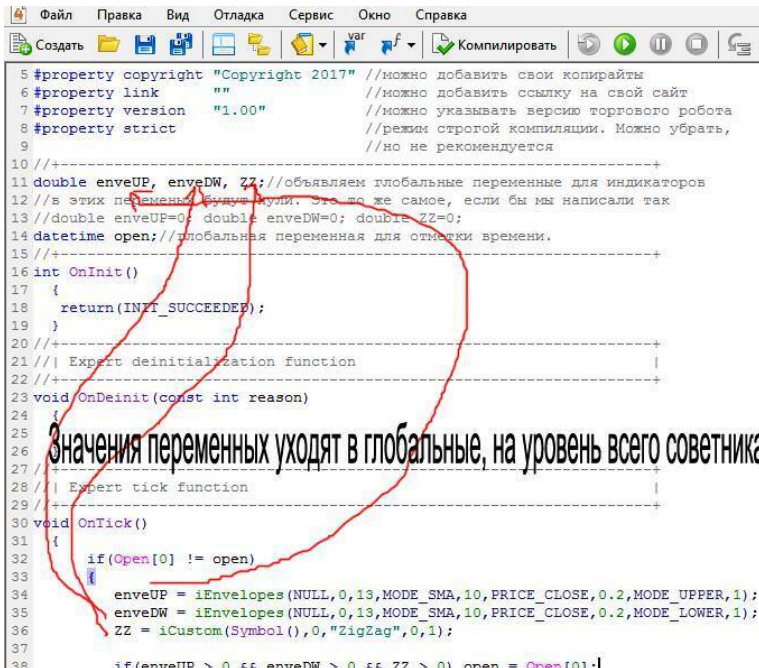
Чтобы узнать ценовые значения индикаторов нам нужно объявить глобальные переменные типа double для верхней и нижней линии индикатора Envelopes. Назовем их enveUP

и enveDW. Эти названия можно придумывать самим. То же самое надо сделать и для получения ценового значения индикатора ZZ. Назовем эту переменную ZZ. Почему именно *глобальные* переменные? Для того, чтобы эти значения мы могли вызывать в любом месте программы (т. е. советника). Дело в том, что мы будем вызывать значения индикаторов не на каждом приходящем тике, а один раз на одной свече. Это существенно повысит производительность, т. к. терминалу не нужно будет выполнять одну и ту же операцию на каждом тике. Если мы обернем в фигурные скобки вызов наших индикаторов с записью их значений НЕ в глобальные переменные, то эти значения будут видны только в рамках этих же фигурных скобок. И за пределами их мы получим ошибку. Более подробно постараюсь описать на рисунке ниже.

Перепишите этот код в свой редактор:

```
//+–+
//| MyFirstEA.mq4 |
//| Copyright 2017, |
//+–+
#property copyright "Copyright 2017"
#property link ""
#property version "1.00"
#property strict
//+–+
double enveUP, enveDW, ZZ;
datetime open;
```

```
//+–+
int OnInit()
{
return(INIT_SUCCEEDED);
}
void OnDeinit(const int reason)
{
}
void OnTick()
{
if(Open[0] != open)
{
enveUP =
iEnvelopes(NULL,0,13,MODE_SMA,10,PRICE_CLOSE,0.2,M
enveDW =
iEnvelopes(NULL,0,13,MODE_SMA,10,PRICE_CLOSE,0.2,M
ZZ = iCustom(Symbol(),0,"ZigZag",0,1);
if(enveUP > 0 && enveDW > 0 && ZZ > 0) open = Open[0];
}
}
```



Разберем, что же означает каждая строчка.

В глобальных переменных, кроме переменных для значений индикаторов, мы объявили переменную типа datetime с названием open. Сейчас она содержит 0.

ВАЖНЫЙ МОМЕНТ: установите курсор на слово datetime и нажмите на клавиатуре F1 – появится СПРАВКА

с описанием, что означает тип datetime. Так можно делать **НА всех** встроенных командах!

if(Open[0] **!=** *open*) : **Если** Время Открытия Нулевой Свечи **НЕ РАВНО** *open* (т. е. нулю), то выполнится код в фигурных скобках. Команда Open[0] означает Время Открытия Нулевой (т. е. текущей, еще не закрытой свечи). Также, установите курсор на Open и нажмите F1 – почитайте, что это за команда.

EnveUP =
iEnvelopes(NULL,0,13,MODE_SMA,10,PRICE_CLOSE,0.2,M
нажимаем на iEnvelopes и видим, в каком порядке и какие данные должны быть указаны:

Конец ознакомительного фрагмента.