

## Программирование на MQL 4: Включаемые файлы, подключаемые модули и импорт функций.

Александр Иванов (AKA HORN)  
<http://forex.tomsk.ru>

Дальнейшее знакомство с языком MQL4 хотелось бы начать с общего рассуждения на тему эффективности трудозатрат программистов. Исходные коды программ, как правило, создаются не с "чистого листа", а в основу практически всех программ ложатся уже имеющиеся наработки. Архитекторы программ давно уже привыкли использовать целые библиотеки заготовок в качестве кирпичиков для создания новых программ. В связи с этим язык программирования, который не позволяет использовать уже имеющиеся наработки, или требующий больших усилий для использования уже однажды выполненной работы, воспринимается как неудобный.

Естественно, что необходимость постоянного повторения одного и того же кода, совершая минимальные изменения, является рутинной задачей и может надолго отбить охоту к творчеству даже у самого терпеливого программиста. Никому не захочется в сотый раз писать блок кода, сортирующий массив. Другое дело, когда есть возможность оформить часто используемые и однажды написанные блоки кода в виде функций. В прошлом номере Forex Magazine мы уже писали о возможностях работы с функциями добавленных в язык программирования MQL4. К счастью, разработчики MetaQuotes Software не ограничились этим новшеством и добавили в MQL 4 ещё как минимум две возможности для повторного использования кода.

Первая из них - возможность включать в файлы своей программы другие файлы. Это позволяет собрать ваши собственные функции, выполняющие родственные действия, в отдельные модули (файлы).

Вторая - это возможность подключать к программе на MQL 4 библиотеки функций, написанных на других языках программирования и оформленных в виде динамически подключаемых библиотек (*dynamically linking libraries - DLL*).

Для того, чтобы использовать первую из упомянутых выше возможностей - включить в файл своей программы

какой-то другой файл, в MQL 4 добавлена директива "#include". Эта идея позаимствована у языка программирования Си. В приведённом ниже примере показано использование директивы "include" для включения в файл вашей программы файла "MyDefines.mq4"

```
#include <MyDefines.mq4>
```

Предположим, что в файле "MyDefines.mq4" мы храним функции, расширяющие функциональность работы со строками языка MQL4. Для наглядности примера предположим так же, что таких функций 20. Не имея директивы "include", нам пришлось бы воспользоваться одним из следующих способов:

- либо скомпилировать каждую из 20-ти функций, как отдельную пользовательскую функцию, сохраняемую в отдельном файле, что, безусловно, внесло бы некоторую неразбериху в работу с библиотекой;

- либо пришлось бы хранить все функции в одном файле, и на этапе написания программы вставлять нужные из них в свою программу с помощью функций копирования/вставки, что ещё менее предпочтительнее, нежели предыдущий вариант. Так как функции могут использовать в процессе работы друг друга, при вставлении в программу некоторой функции нам пришлось бы внимательно проследить ссылки и скопировать все функции используемые интересующей нас функцией.

Директива "include" избавляет нас от обеих описанных проблем. К тому же, компилятор языка MQL 4 достаточно умён, чтобы выбрать из включённого файла только те функции, которые могут быть вызваны в процессе работы скомпилированной программы. Это гарантирует, что программа не будет беспринципно увеличиваться после добавления каких-либо новых функций в подключаемый файл библиотеки, если, конечно, они не будут прямо или косвенно использоваться в коде.

# FOREX MAGAZINE

Программирование на MQL 4: Подключаемые модули и импорт функций

Июнь 2004 №23

Рассмотрим вторую возможность повторного использования кода, причём в этом случае мы уже не ограничены только собственными наработками. Любая DLL-библиотека с экспортруемыми функциями может стать объектом нашего пристального внимания.

Для подключения к программе на MQL 4 библиотек функций, написанных на других языках программирования в виде DLL, следует воспользоваться директивой "#import". Пример показывает, как из стандартной, поставляемой с Windows, библиотеки user32.dll импортировать функцию MessageBeep():

```
#import "user32.dll"
    int MessageBeep(int uType);
#import
```

Думаю, что многие программисты уже поняли, что таким образом появилась возможность обращаться практически к любой функции Win32 API. Так же эта возможность позволяет самостоятельно написать на любом более или менее развитом языке программирования собственную DLL-библиотеку, в которой могут быть реализованы наши самые смелые мечты.

Кстати, первая из рассмотренных возможностей прекрасно дополняет вторую. Мы, конечно, могли бы включать в наши программы импорт библиотечных функций по необходимости, но всё же, для дальнейшего использования, было бы, наверное, удобнее раз и навсегда "завернуть" в отдельный файл импорт функций из одной библиотеки и включать его в программы директивой "include" всякий раз, когда в этом будет необходимость.

Предположим, что нас интересуют некоторые функции, хранящиеся в файле MyLib.dll, которые часто используются при написании нескольких индикаторов, которые, в свою очередь, использованы в эксперте. Единожды написав файл, например, MyLib.mq4, в котором импортируются всевозможные функции из MyLib.dll, мы облегчим себе работу. Теперь нам не придётся импортировать по отдельности каждую функцию из DLL-библиотеки. Нужно будет просто включить в нашу программу файл MyLib.mq4 и

пользоваться любой из скрытых в MyLib.dll экспортруемых функций. Файл MyLib.mq4 будет содержать следующие строки:

```
// Это содержимое файла MyLib.mq4
#import "MyLib.dll"
    int Function1(int uParam1);
    int Function2(int uParam1);
    int Function3(int uParam1);
    int Function4();
    int Function5(int uParam1,
                  int uParam2);
#import
```

Тогда программа, использующая функцию Function5 из MyLib.dll, сможет сделать это так:

```
// Подключить MyLib.mq4 к программе
#include <MyLib.mq4>

int init()
{
    int nRet = 0;
    // do something...
    nRet = Function5(10, 20);
    // do something else...
    return(0);
}
```

Вот на сегодня и всё, о чём хотелось бы рассказать. После выпуска программы Meta Trader 4, мы обязательно вернёмся к этой интересной теме и проиллюстрируем вышеописанное на конкретных примерах.

Александр Иванов  
для Forex Magazine