

## Программирование на MQL 4: Оператор выбора "switch"

Александр Иванов (AKA HORN)

<http://www.fxtrade.tomsk.ru>

На прошлой неделе ожидания многих из нас были частично удовлетворены - компания Meta Quotes Software выложила на своём сайте (<http://www.metaquotes.ru>) бета-версию редактора программ MetaEditor 4, входящего в состав новой версии информационно-торговой системы MetaTrader 4. Публичное представление редактора MetaEditor 4 позволяет, что называется, "вживую" познакомиться с языком программирования MQL 4. Теперь нам предоставилась возможность оценить набор функций, доступных программисту для написания собственных индикаторов и советников.

Будем надеяться, что вскоре у нас появится возможность поэкспериментировать не только с языком программирования, но и увидеть на торговом терминале комплекса MetaTrader 4 написанные программы в действии.

Теперь отвлечёмся от этой, безусловно, волнующей новости и вернёмся к изучению MQL 4. Перейдём к рассмотрению оператора выбора "switch".

В прошлом номере журнала Forex Magazine были рассмотрены условные операторы "if". Там же было отмечено, что следует строить алгоритмы своих программ так, чтобы избежать использования второй части "else" оператора "if". Но на практике не всегда удаётся этого достичь, более того часто бывают ситуации, когда в условии оператора "if" нельзя однозначно поставить выражение, дающее один из двух результатов "true" или "false".

Проще объяснить это на примере. Допустим, нам известно, что некоторая переменная принимает несколько чётко определённых значений. Нам следует проверить, равна ли эта переменная каждому из предопределённых значений, и, в зависимости от результата проверки, выполнить разные действия. Видно, что оператор, который позволяет выполнить одно из двух действий (в зависимости от значения условия) здесь не совсем подходит. То есть, когда приходится делать множественный выбор, классический вариант

оператора "if" - не лучшее решение. Можно, конечно, использовать серию вложенных операторов "if", которая будет выглядеть следующим образом:

```
if(my_var == condition1) {  
  
    // если выполнено первое  
    // условие, то выполнить  
    // этот блок  
  
} else {  
  
    if(my_var == condition2) {  
  
        // если выполнено второе  
        // условие, то выполнить  
        // этот блок  
  
    } else {  
  
        if(my_var == condition3) {  
  
            // если выполнено третье  
            // условие, то выполнить  
            // этот блок  
  
        } else {  
            // если не выполнено  
            // ни одно из трёх  
            // предыдущих условий, то  
            // выполнить этот блок  
        }  
    }  
}
```

Видно, что в случае, когда переменная `my_var` проверяется на совпадение лишь с тремя различными значениями, использование оператора "if" вносит некоторую суматоху в код программы: во-первых, очень легко запутаться в расставленных скобках, во-вторых, код становится громоздким, трудным для понимания, и, к тому же, поиск ошибки в таком коде может доставить много хлопот. Представьте себе, что будет твориться в том случае, когда переменная будет проверяться на совпадение с десятью различными значениями.

К счастью, для таких случаев в MQL 4 добавлен оператор "switch". В применении к рассмотренной выше ситуации использование оператора "switch" выглядит так:

```
switch(my_var) {  
  
    case condition1: {  
        // если выполнено первое  
        // условие, то выполнить  
        // этот блок  
    } break;  
  
    case condition2: {  
        // если выполнено второе  
        // условие, то выполнить  
        // этот блок  
    } break;  
  
    case condition3: {  
        // если выполнено третье  
        // условие, то выполнить  
        // этот блок  
    } break;  
  
    default: {  
        // если не выполнено  
        // ни одно из трёх  
        // предыдущих условий, то  
        // выполнить этот блок  
    }  
}
```

Этот код выглядит более лёгким для восприятия. А теперь поясним синтаксис оператора "switch", и как этот оператор работает.

Сразу за ключевым словом "switch" в круглых скобках следует переменная, значение которой будет сравниваться с несколькими значениями. Следом за этим, в операторных скобках заключены параметры "case" (один или больше), каждый из которых содержит проверяемое значение. После того, как программа переходит к выполнению конструкции "switch ... case", происходит последовательное сравнение переменной, указанной после ключевого слова "switch" со значениями, следующими за ключевыми словами "case".

В самом конце сравнения может присутствовать необязательный параметр "default". Если программа не выполнит ни один из блоков "case", то она переходит к блоку "default". Его следует воспринимать, как блок, выполняющийся по умолчанию, то есть выполняющийся при отсутствии каких-либо блоков "case", обрабатывающих сложившуюся ситуацию.

Обратите внимание, что каждый блок "case" заканчивается ключевым словом "break". Если не поставить в конце обработки блока "case" оператор "break", то будут выполняться последующие блоки "case" и "default" уже без каких-либо проверок и до тех пор, пока не встретится "break" или не закончится область действия оператора "switch". Этот нюанс имеет своё применение в тех случаях, когда нескольким значениям проверяемой переменной следует сопоставить один и тот же обработчик.

```
switch(my_var) {  
  
    case condition1:  
    case condition2: {  
        // выполнить этот блок,  
        // если выполнено первое  
        // или второе условие.  
    } break;  
  
    case condition3: {  
        // выполнить этот блок,  
        // если выполнено третье  
        // условие.  
    } break;  
}
```

Благодаря тому, что MetaEditor 4 стал доступен широким массам, мы можем заглянуть в примеры программ, поставляемых в комплекте с ним. Использование оператора "switch" проиллюстрировано в индикаторе "Moving Average 4". Следующий код показывает, как для вычисления различных типов МА используется рассмотренный нами в этой статье оператор "switch".

```
// calculate current moving
// average value

switch(g_MAType) {
    case 0:
        CalculateSMA();
        break;
    case 1:
        CalculateEMA();
        break;
    case 2:
        CalculateSMMA();
        break;
    case 3:
        CalculateLWMA();
        break;
}
```

Предлагаем внимательно изучить все доступные коды программ, чтобы на практике усвоить материал серии статей по программированию на MQL 4.

На сегодня всё. Желаящие задать вопросы по программированию на MQL или просто поговорить на околофинансовые темы, могут поучаствовать в дискуссиях форума на сайте <http://fxtrade.tomsk.ru>

*Александр Иванов  
для Forex Magazine*