MQL4 COURSE

By Coders' guru
www.forex-tsd.com

-6-

Loops & Decisions

Part 2

Welcome to the sixth lesson in my course about MQL4.
I hope you enjoyed the previous lessons.

In the previous lesson, we have talked about the Loops.
And we have seen that the Loops are one of two ways we use to change the normal flow of the program execution -from top to bottom. The second way is the Decisions.

Decisions in a program cause a one-time jump to a different part of the program, depending on the value of an expression.
These are the kinds of decisions statements available in MQL4:

The if Statement

The **if** statement is the simplest decision statement, here's an example:

```
if( x < 100 )
Print("hi");
```

Here the **if** keyword has followed by parentheses, inside the parentheses the **Test expression (x < 100)**, when the result of test expression is **true** the body of the **if** will execute **(Print("hi");)** ,and if it is false, the control passes to the statement follows the **if** block.

Figure 1 shows the flow chart of the **if** statement**:**

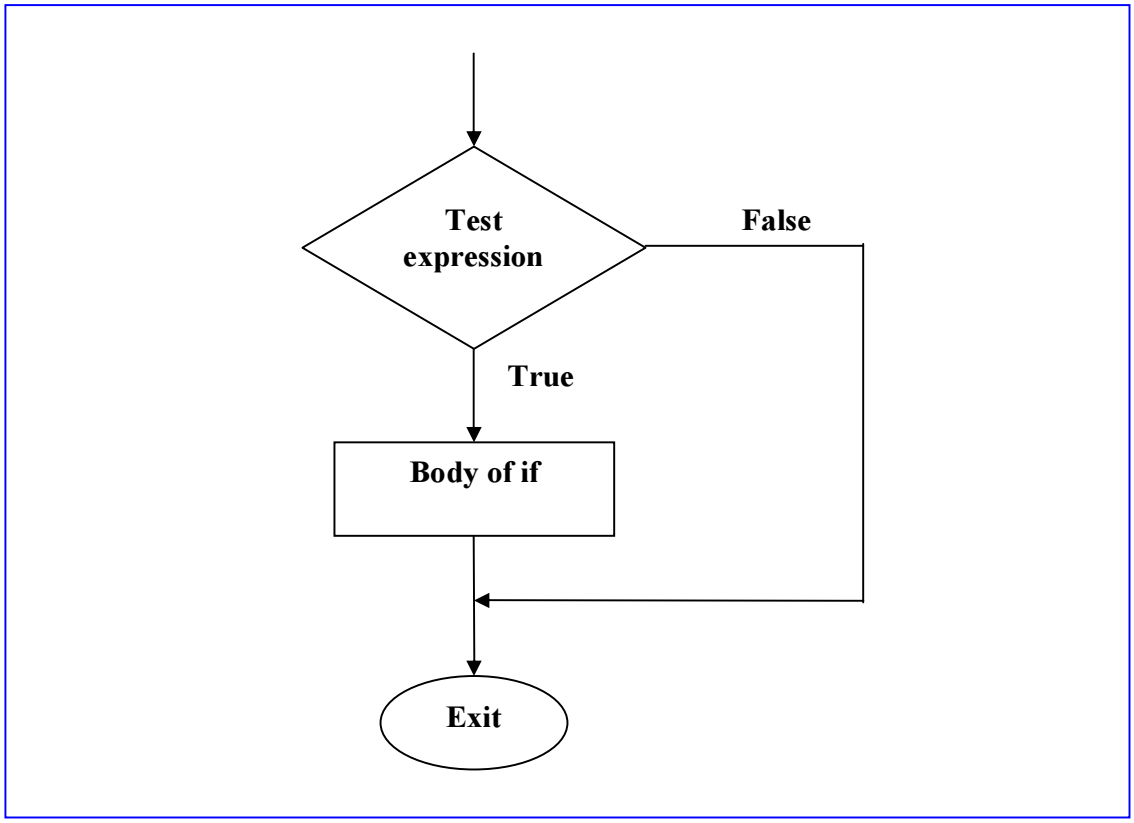

Figure 1 - Flow chart of the if statement

Multi Statements in the if Body:

Like the loops, the body of **if** can consist of more than statement delimited by braces.

For example:

```
if(current_price==stop_lose)
{
        Print("you have to close the order");
        PlaySound("warning.wav");
}
```

Notice the symbol == in the Test expression; it's one of the Relational Operators you have studied in the lesson 4, operations & expressions.
This is a source of a lot of errors, when you forget and use the assignment operator =.

Nesting:

The loops and decision structures can be basted inside one another; you can nest **ifs**

inside loops, loops inside **ifs, ifs** inside **ifs**, and so on.

Here's an example:

```
for(int i=2 ; i<10 ; i++)
        if(i%2==0)
        {
                Print("It's not a prime nomber");
                PlaySound("warning.wav");
        }
```

In the previous example the **if** structure nested inside the for loop.

*Notice: you will notice that there are no braces around the loop body, this is because the **if** statement and the statements inside its body, are considered to be a single statement.*

The if...else Statement

The **if** statement let's you to do something if a condition is true, suppose we want to do another thing if it's false. That's the **if...else** statement comes in.
It consist of **if** statement followed by statement or a block of statements, then the **else** keyword followed by another statement or a block of statements.

Like this example:

```
if(current_price>stop_lose)
        Print("It’s too late to stop, please stop!");
else
        Print("you playing well today!");
```

If the test expression in the **if** statement is true, the program one message, if it isn’t true, it prints the other.

Figure 2 shows the flow chart of the **if…else** statement**:**

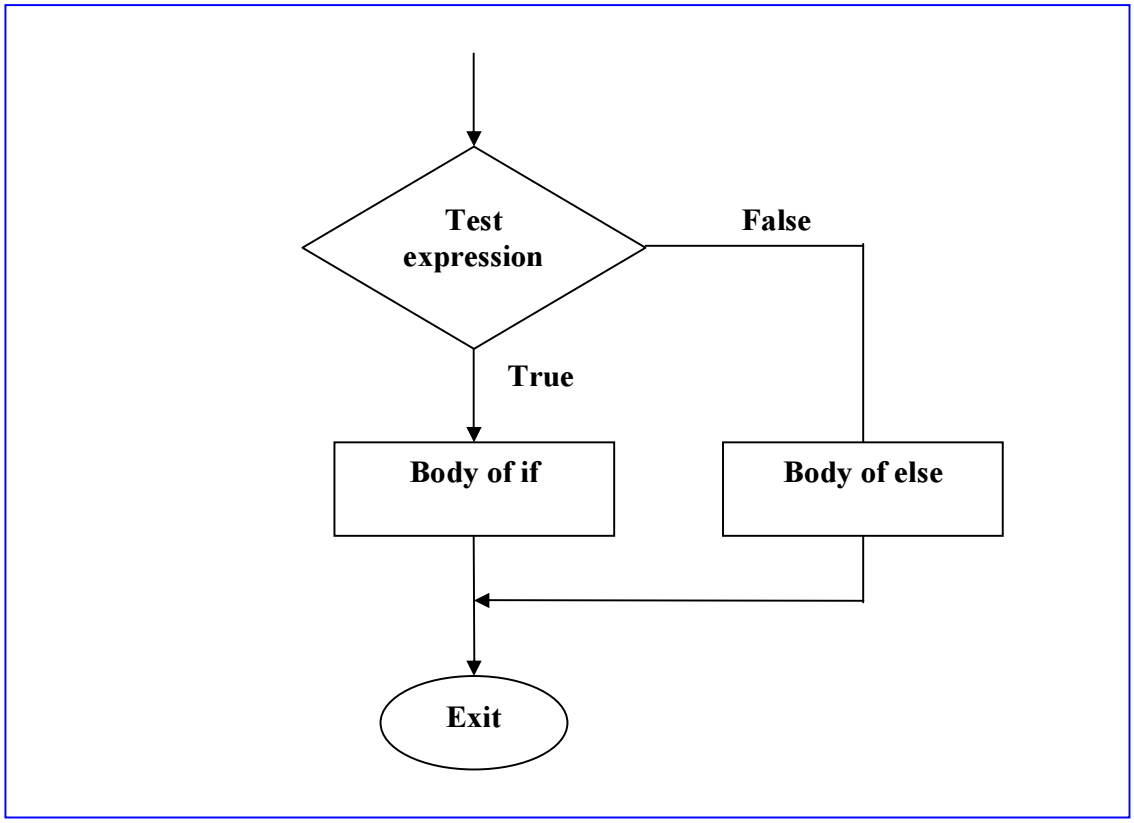

Figure 2 - Flow chart of the if..else statement

Nested if…else Statements

You can nest if…else statement in ifs statements, you can nest if…else statement in if…else statement, and so on.

Like this:

```
if(current_price>stop_lose)
        Print("It's too late to stop, please stop!");
if(current_price==stop_lose)
        Print("It's time to stop!");
else
        Print("you playing well today!");
```

There's a potential problem in nested if…else statements, you can inadvertently match an else with the wrong if.

To solve this case you can do one of two things:

1- you can delimited the **if…else** pairs with braces like this:

```
if(current_price>stop_lose)
{
        Print("It's too late to stop, please stop!");
if(current_price==stop_lose)
        Print("It's time to stop!");
else
        Print("you playing well today!");
}
```

2- If you can't do the first solution (in the case of a lot of if…else statements or you are lazy to do it) take it as rule.
Match else with the nearest if. (Here it's the line **if(current_price==stop_lose)**).

The switch Statement

If you have a large decision tree, and all the decisions depend on the value of the same variable, you can use a switch statement here.
Here's an example:

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

In the above example the switch keyword is followed by parentheses, inside the parentheses you'll find the **switch constant**, this constant can be an integer, a character constant or a constant expression. The constant expression mustn't include variable for example:
case X+Y: is invalid switch constant.

How the above example works?

The switch statement matches the constant **x** with one of the **cases constants**.
In the case **x=='A'** the program will print **"CASE A"** and the **break** statement will take you the control out of the switch block.

In the cases **x=='B'** or **x=='C'**, the program will print **"CASE B or C"**. That's because there's no **break** statement after **case 'B':**.

In the case that **x !=** any of the cases constants the switch statement will execute the **default** case and print **"NOT A, B or C".**

Figure 3 shows the flow chart of the switch statement

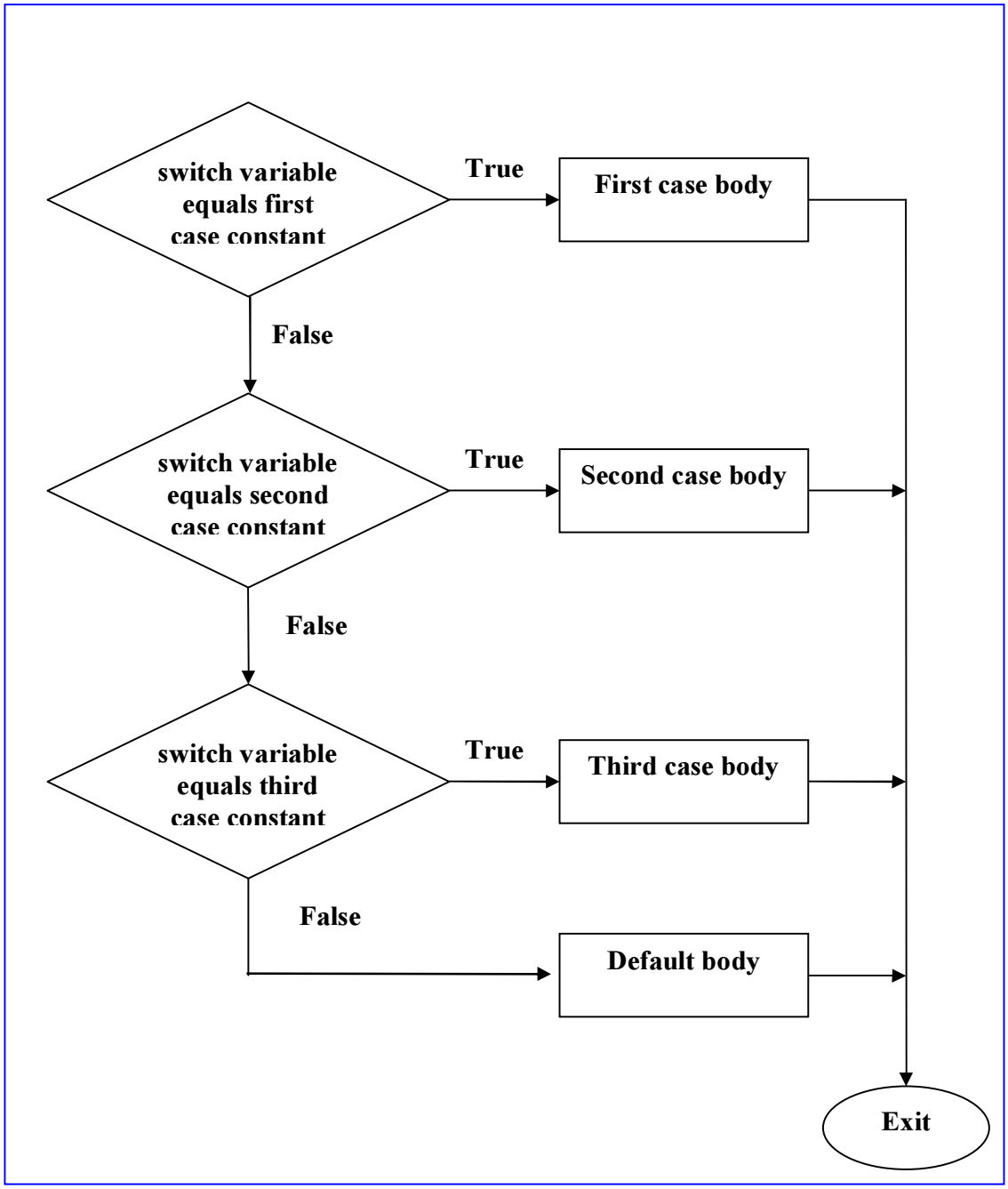

Figure 3 - Flow chart of the switch statement

I hope you enjoyed the lesson.
I welcome very much the questions and the suggestions.

See you
Coders' Guru
25-10-2005