MQL4 COURSE

By Coders' guru
www.forex-tsd.com

-5-

Loops & Decisions
Part 1

Welcome to the fifth lesson in my course about MQL4.

You can download the previous lesson from here:
http://forex-tsd.com/attachment.php?attachmentid=399
http://forex-tsd.com/attachment.php?attachmentid=372
http://forex-tsd.com/attachment.php?attachmentid=469
http://forex-tsd.com/attachment.php?attachmentid=481
Don't forget to login first.

The normal flow control of the program you write in MQL4 (And in others languages as well) executes from top to bottom, A statement by a statement.

A statement is a line of code telling the computer to do something.
For example:
Print("Hello World");
return 0;
A semicolon at end of the statement is a crucial part of the syntax but usually easy to forget, and that's make it the source of 90% of errors.

But the top bottom execution is not the only case and it has two exceptions,
They are the loops and the decisions.

The programs you write like -the human- decides what to do in response of circumstances changing. In these cases the flow of control jumps from one part of the program to another.
Statements cause such jumps is called Control Statements.
Such controls consist of Loops and Decisions.

LOOPS

Loops causing a section of your program to be repeated a certain number of times.
And this repetition continues while some condition is true and ends when it becomes false.
When the loop end it passes the control to next statement follow the loop section.

In MQL4 there are two kinds of loops:

The for Loop

The **for** loop considered the easiest loop because all of its control elements are gathered in one place.
The **for** loop executes a section of code a fixed number of times.

For example:

```
int j;
for(j=0; j<15; j++)
        Print(j);
```

How does this work?

The **for** statement consists of **for** keyword, followed by parentheses that contain three expressions separated by semicolons:

for(j=0; j<15; j++)

These three expressions are the **initialization** expression, the **test** expression and the **increment** expression:

j=0 ← initialization expression
j<15 ← test expression
J++ ← increment expression

The **body** of the loop is the code to be executed the fixed number of the loop:

Print(j);

This executes the body of the loop in our example for 15 times.

Note: the **for** statement in not followed by a semicolon. That's because the **for** statement

and the loop body are together considered to be a program statement.

The initialization expression:

The initialization expression is executed only once, when the loop first starts. And its purpose to give the loop variable an initial value (0 in our example).

You can declare the loop variable outside (before) the loop like our example:

int j;

Or you can make the declaration inside the loop parentheses like this:

for(int j=0; j<15; j++)

The previous two lines of code are equal, except the **Scope** of each variable (you will know more about the variable declaration and scopes in the Variables lesson).
The outside declaration method makes every line in the code block to know about the variable, while the inside declaration makes only the **for** loop to know about the variable.

You can use more that one initialization expression in **for** loop by separating them with comma (,) like this:

```
int i;
int j;
for(i=0 ,j=0;i<15;i++)
  Print(i);
```

The Test expression:

The test expression always a relational expression that uses relational operators (please refer to relational operators in the previous lesson).

It evaluated by the loop every time the loop executed to determine if the loop will continue or will stop. It will continue if the result of the expression is true and will stop if it false.

In our example the **body** loop will continue printing **i** (**Print(i)**) while the case **j<15** is true. For example the j = 0,1,2,3,4,5,6,7,8,9,10,11,12,13 and 14.
And when **j** reaches **15** the loop will stops and the control passes to the statement following the loop.

The Increment expression:

The increment expression changes the value of the loop variable (**j** in our example) by increase it value by 1.
It executed as the last step in the loop steps, after initializing the loop variable, testing the test expression and executing the body of the loop.

Figure 1 shows a flow chart of the **for** loop.

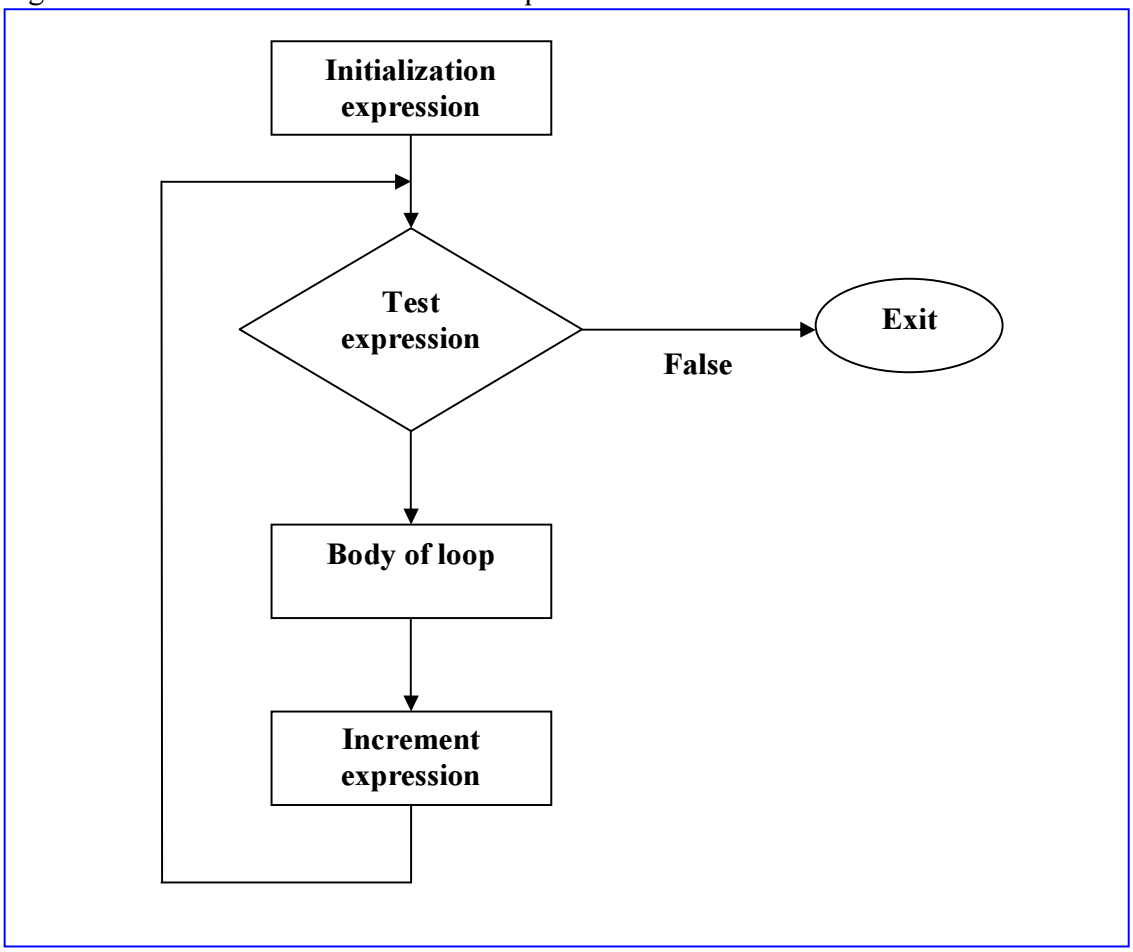

Figure 1 - Flow chart of the for loop

Like the initialization expression, in the increment expression you can use more than one increment expression in the **for** loop by separating them with comma (,) like this:

```
int i;
int j;
for(i=0 ,j=0;i<15,i<;i++,j++)
  Print(i);
```

But you can only use one test expression.

Another notice about the increment expression, it's not only can increase the variable of the loop, but it can perform and operation it like for example decrements the loop variable like this:

```
int i;
for(i=15;i>0,i<;i--)
  Print(i);
```

The above example will initialize the **i** to **15** and start the loop, every time it decreases **i** by **1** and check the test expression (**i>0**).
The program will produce these results: 15,14,13,12,11,10,9,8,7,6,5,4,3,2,1.

Multi statement in the loop body:

In our previous examples, we used only one statement in the body of the loop, this is not always the case.
You can use multi statements in the loop body delimited by braces like this:

```
for(int i=1;i<=15;i++)
{
  Print(i);
  PlaySound("alert.wav");
}
```

In the above code the body of the loop contains two statements, the program will execute the first statement then the second one every time the loop executed.
Don't forget to put a semicolon at the end of every statement.

The Break Statement:

When the keyword presents in the **for** loop (and in **while** loop and **switch** statement as well) the execution of the loop will terminate and the control passes to the statement followed the loop section.

For example:

```
for(int i=0;i<15;i++)
{
 if((i==10)
   break;
 Print(i);
}
```

The above example will execute the loop until **i** reaches **10**, in that case the break keyword

will terminate the loop. The code will produce these values: 0,1,2,3,4,5,6,7,8,9.

The Continue Statement:

The break statement takes you out the loop, while the continue statement will get you back to the top of the loop (parentheses).
For example:

```
for(int i=0;i<15; i++)
{
  if(i==10) continue;
  Print(i)
}
```

The above example will execute the loop until **i** reaches **10**, in that case the continue keyword will get the loop back to the top of the loop without printing **i** the tenth time. The code will produce these values: 0,1,2,3,4,5,6,7,8,9,11,12,13,14.

Latest note:

You can leave out some or all of the expressions in **for** loop if you want, for example:
for(;;)
This loop is like **while** loop with a test expression always set to true.
We will introduce the **while** loop to you right now.

The while Loop

The **for** loop usually used in the case you know how many times the loop will be executed.
What happen if you don't know how many times you want to execute the loop?
This the **while** loop is for.

The **while** loop like the **for** loop has a Test expression. But it hasn't Initialization or Increment expressions.

This is an example:

```
int i=0;
while(i<15)
{
  Print(i);
  i++;
```

```
}
```

In the example you will notice the followings:

- The loop variable had declared and initialized before the loop, you can not declare or initialize it inside the parentheses of the **while** loop like the **for** loop.
- The **i++** statement here is not the increment expression as you may think, but the body of the loop must contain some statement that changes the loop variable, otherwise the loop would never end.

How the above example does work?

The **while** statement contains only the Test expression, and it will examine it every loop, if it's true the loop will continue, if it's false the loop will end and the control passes to the statement followed the loop section.

In the example the loop will execute till **i** reaches **16** in this case **i<15=false** and the loop ends.

Figure 2 shows a flow chart of the **while** loop.

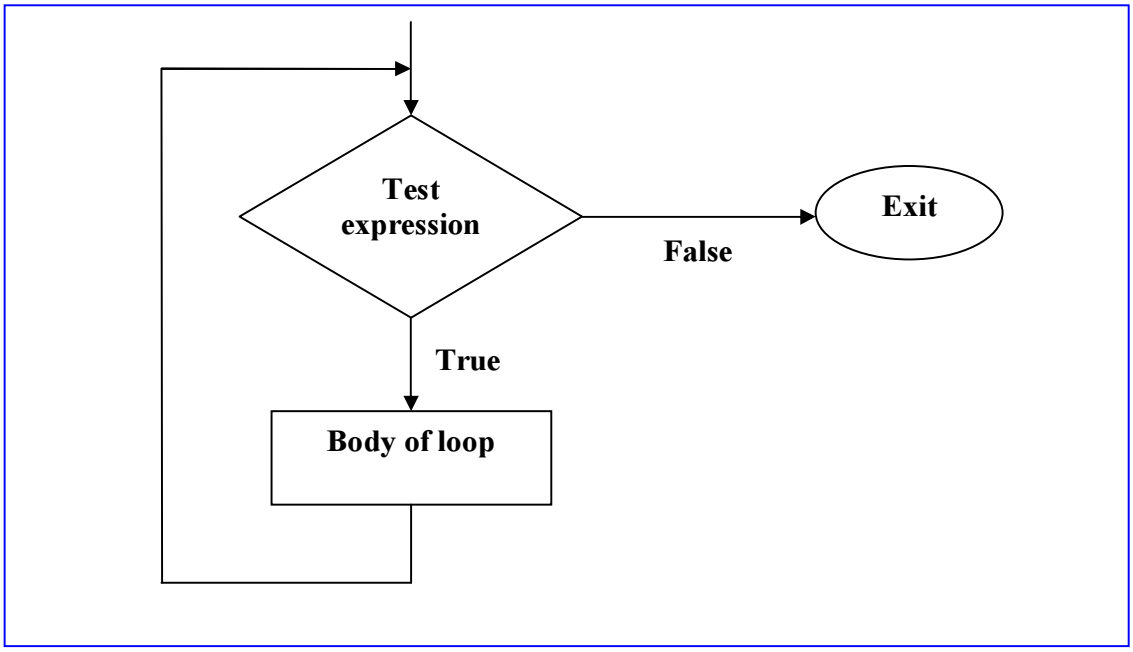

Figure 2 - Flow chart of the while loop

I told you before that the **while** loop is like the **for** loop, these are the **similar aspects**:

1. You can use **break** statement and **continue** in both of them.
2. You can single or multi statements in the body of the loop in both of them, in the case of using multi statements you have to delimit them by braces.
3. The similar copy of **for(;;)** is **while(true)**

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

See you
Coders' Guru
24-10-2005