MQL4 COURSE

By Coders' guru
www.forex-tsd.com

-4-

Operations & Expressions

Welcome to the fourth lesson in my course about MQL4.
The previous lesson "Data Types" presented a lot of new concepts; I hope you understand it, and above all you **enjoyed** it.

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
Don't forget to login first.

Now, let's enjoy the Operations & Expressions.

What's the meaning of Operations & Expressions?

You know the operations very well. If I told you that (+,-,*, /) are the basic arithmetical operators, you will remember very fast what's the operator means.

I hear you saying "OK, I know the operations; could you tell me what's the meaning of the expression?"
Identifiers (do you remember them? If not, Review the SYNTAX lesson) together with the **Operations** produce the **Expressions**.
Puzzled? Let's illustrate it in an example:

```
x = (y*z)/w;
```

x,**y**,**z** and **w**, here are **identifiers**.
=,* and / are the **operators**.
The whole line is an **expression**.

*When the expressions combined together it makes a **statement.***
*And when the statements combined together it makes a **function** and when the functions combined together it makes a **program**.*

In the remaining of this lesson we are going to talk about the kinds operators used in MQL4.

So, let's start with the basic arithmetical operators:

1- Arithmetical operators:

In MQL4 there are 9 Arithmetical operations
This is the list of them with the usage of each:

Operator	Name	Example	Description
+	Addition operator	**A = B + C;**	Add **B** to **C** and assign the result to **A**.
-	Subtraction operator	**A = B - C;**	Subtract **C** from **B** and assign the result to **A**.
+ -	Sign changer operators	**A = -A;**	Change the sign of **A** from positive to negative.
*	Multiplication operator	**A = B * C;**	Multiply **B** and **C** and assign the result to **A**.
/	Division operator	**A = B / C;**	Divide **B** on **C** and assign the result to **A**.
%	Modulus operator	**A =A % C;**	**A** is the reminder of division of **B** on **C**. (ex: 10%2 will produce 0, 10%3 will produce 1).
++	Increment operator	**A++;**	Increase **A** by **1** (ex: if A =1 make it 2).
--	Decrement operator	**A--;**	Decrease **1** from **A** (ex: if A =2 make it 1).

Note: The remainder operator works by dividing the first number by the second number for the first integer results and then returns the remaining number.

For example:
10%5=0
This is because if you divide 10 by 5 you will get 2 and there no remaining value, so the remainder is 0.
10%8=2
This is because if you divide 10 by 8 you will get 1 (1*8=8), so the remainder is (10-8 = 2).
100%15=10

This is because if you divide 100 by 15 you will get 6 (6*15=90), so the remainder is (100-90=10).
What about 6%8?
It will be 6 because if you divide 6 by 8 you will get 0 (8*0=0), so the remainder is (6-0=6).

Note: You can't combine the increment and decrement operator with other expressions. For example you can't say:

```
A=(B++)*5;
```

But you can write it like that:

```
A++;
B=A*5;
```

Note: How the above example works? Let's assume:
int A=1; //set A to 1
int B;
A++; //increase A by 1, now A=2
B=A*5; //which means B=2*5

2- Assignment operators:

The purpose of any expression is producing a result and the assignment operators setting the left operand with this result.

For example:

```
A = B * C;
```

Here we multiply **B** and **C** and assign the result to **A**.
(=) here is the assignment operator.

In MQL4 there are 11 assignments operations
This is the list of them with the usage of each:

Operator	Name	Example	Description
=	Assignment operator	**A = B;**	Assign **B** to **A.**
+=	Additive Assignment operator	**A += B;**	It's equal to: **A = A + B;** Add **B** to **A** and assign the result to **A**.

-=	Subtractive Assignment operators	**A -= B;**	It's equal to: **A = A - B;** Subtract B from A and assign the result to A.
***=**	Multiplicative Assignment operator	**A *= B;**	It's equal to: **A = A * B;** Multiply A and B and assign the result to A.
/=	Divisional Assignment operator	**A /= B;**	It's equal to: **A = A / B;** Divide A on B and assign the result to A.
%=	Modulating Assignment operator	**A %= B;**	It's equal to: **A = A % B;** Get the reminder of division of **A** on **B** and assign the result to **A**.
>>=	Left Shift Assignment operator	**A >>= B;**	It shifts the bits of **A** left by the number of bits specified in **B**.
<<=	Right Shift Assignment operator	**A <<= B;**	It shifts the bits of **A** right by the number of bits specified in **B**.
&=	AND Assignment operator	**A &= B;**	Looks at the binary representation of the values of **A** and **B** and does a bitwise AND operation on them.
|=	OR Assignment operator	**A |= B;**	Looks at the binary representation of the values of **A** and **B** and does a bitwise OR operation on them.
^=	XOR Assignment operator	**A ^= B;**	Looks at the binary representation of the values of two **A** and **B** and does a bitwise exclusive OR (XOR) operation on them.

3- Relational operators:

The relational operators compare two values (operands) and result false or true only.
It's is like the question "Is **John** taller than **Alfred**? Yes / no?"
The result will be false only if the expression produce zero and true if it produces any number differing from zero;

For example:

```
4 == 4;          //true
4 < 4;           //false
4 <= 4          //true;
```

In MQL4 there are 6 Relational operations
This is the list of them with the usage of each:

Operator	Name	Example	Description
==	Equal operator	**A == B;**	True if **A** equals **B** else False.
!=	Not Equal operator	**A != B;**	True if **A** does not equal **B** else False.
<	Less Than operators	**A < B;**	True if **A** is less than **B** else False.
>	Greater Than operator	**A > B;**	True if **A** is greater than **B** else False.
<=	Less Than or Equal operator	**A <= B;**	True if **A** is less than or equals **B** else False.
>=	Greater Than or Equal operator	**A >= B;**	True if **A** is greater than or equals **B** else False.

4- Logical operators:

Logical operators are generally derived from Boolean algebra, which is a mathematical way of manipulating the truth values of concepts in an abstract way without bothering about what the concepts actually *mean*. The truth value of a concept in Boolean value can have just one of two possible values: true or false.
MQL4 names the Logical operators as Boolean operators

MQL4 uses the most important 3 logical operators.
This is the list of them with the usage of each:

Operator	Name	Example	Description
&&	AND operator	**A && B;**	If either of the values are zero the value of the expression is zero, otherwise the value of the expression is 1. If the left hand value is zero, then the right hand value is not considered.
||	OR operator	**A || B;**	If both of the values are zero then the value of the expression is 0 otherwise the value of the expression is 1. If the left hand value is non-zero, then the right hand value is not considered.
!	NOT operator	**!A;**	Not operator is applied to a non-

			zero value then the value is zero, if it is applied to a zero value, the value is 1.

5- Bitwise operators:

The bitwise operators are similar to the logical operators, except that they work on a smaller scale -- binary representations of data.

The following operators are available in MQL4:

Operator	Name	Example	Description
&	AND operator	**A & B;**	Compares two bits and generates a result of 1 if both bits are 1; otherwise, it returns 0.
|	OR operator	**A | B;**	Compares two bits and generates a result of 1 if the bits are complementary; otherwise, it returns 0.
^	EXCLUSIVE-OR operator	**A ^ B;**	Compares two bits and generates a result of 1 if either or both bits are 1; otherwise, it returns 0.
~	COMPLEMENT operator	**~A;**	Used to invert all of the bits of the operand.
>>	The SHIFT RIGHT operator	**A >> B;**	Moves the bits to the right, discards the far right bit, and assigns the leftmost bit a value of 0. Each move to the right effectively divides `op1` in half.
<<	The SHIFT LEFT operator	**A << B;**	Moves the bits to the left, discards the far left bit, and assigns the rightmost bit a value of 0. Each move to the left effectively multiplies `op1` by 2.

Note Both operands associated with the bitwise operator must be integers.

6- Other operators:

There are some operators which used in MQL4 and don't belong to one of the previous categories:

1- The array indexing operator (**[]**).
2- The function call operator (**()**);
3- The function arguments separator operator -comma (**,**)

We will know more about the Arrays and Functions in the next lessons, so just remember these 3 operators as "Other operators".

Operators Precedence:

If you don't explicitly indicate the order in which you want the operations in a compound expression to be performed, the order is determined by the precedence assigned to the operators in use within the expression. Operators with a higher precedence get evaluated first. For example, the division operator has a higher precedence than does the addition operator. Thus, the two following statements are equivalent:

```
x + y / 100
x + (y / 100) //unambiguous, recommended
```

When writing compound expressions, you should be explicit and indicate with parentheses **()** which operators should be evaluated first. This practice will make your code easier to read and to maintain.

The following table shows the precedence assigned to the operators in the MQL4. The operators in this table are listed in precedence order: The higher in the table an operator appears, the higher its precedence. Operators with higher precedence are evaluated before operators with a relatively lower precedence. Operators on the same **group** have equal precedence. When operators of equal precedence appear in the same expression, a rule must govern which is evaluated first. All binary operators except for the assignment operators are evaluated from left to right. Assignment operators are evaluated right to left.

()	Function call	*From left to right*
[]	Array element selection	

!	Negation	*From left to right*

~	Bitwise negation	
-	Sign changing operation	
*	Multiplication	*From left to right*
/	Division	
%	Module division	
+	Addition	*From left to right*
-	Subtraction	
<<	Left shift	*From left to right*
>>	Right shift	
<	Less than	*From left to right*
<=	Less than or equals	
>	Greater than	
>=	Greater than or equals	
==	Equals	*From left to right*
!=	Not equal	
&	Bitwise AND operation	*From left to right*
^	Bitwise exclusive OR	*From left to right*
&&	Logical AND	*From left to right*
||	Logical OR	*From left to right*
=	Assignment	*From right to left*
+=	Assignment addition	
-=	Assignment subtraction	
*=	Assignment multiplication	
/=	Assignment division	
%=	Assignment module	

>>=	Assignment right shift
<<=	Assignment left shift
&=	Assignment bitwise AND
|=	Assignment bitwise OR
^=	Assignment exclusive OR

,	Comma	*From left to right*

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

See you
Coders' Guru
23-10-2005