ج- العمليات والتعبيرات

بعض الحروف ومتسلسلات الحروف لها اهمية خاصه مثل :

```
+ - * / %      symbols of arithmetic operations
&& ||          symbols of logic operations
= += *=        symbols of assignment operations
```

رموز العمليات هذه تستخدم فى التعبيرات

علامات الترقيم لها أهمية أيضاً مثل (الأقواس) والمحدد } والفاصلة ، وعلامات التنصيص : و ؛

تستخدم رموز العمليات وعلامات الترقيم والمسافات لفصل عناصر لغة البرمجه بين بعضها البعض

وتنقسم التعبيرات والتغيرات الى :

* التعبيرات
* العمليات الحسابية
* عمليات المساواة
* عمليات المقارنة
* العمليات المنطقية
* العمليات الأخرى
* القواعد الأسبقية

التعبيرات :

يتكون من معاملة او اكثر بالاضافة للحروف المستخدمة فى العمليات

ويمكن كتابتة على اكثر من سطر

مثال :

```
a++; b = 10;
x = (y * z) /
    (w + 2) + 127;
```

التعبير الذى ينتهى بالعلامة (;) يعتبر متحكم وليس تعبير

العمليات الحسابية :

تتضمن العمليات الحسابية عمليات الجمع والاضافة وعمليات المضاعفة

```
Sum of values                              i = j + 2;
Difference of values                       i = j - 3;
Changing the operation sign                x = - x;
Product of values                          z = 3 * x;
Division quotient                          i = j / 5;
Division remainder                         minutes = time % 60;
Adding 1 to the variable value             i++;
Subtracting 1 from the variable value      k--;
```

عملية اضافة أو طرح الرقم واحد (جمع او طرح) لا يمكن ان تستخدم فى التعبيرات

مثال :

```
int a=3;
a++;              // valid expression
int b=(a++)*3;    // invalid expression
```

عمليات المساواة :

وهى أن القيمة التى تخص التعبير الذى يحتوى على العملية المعطاة هى نفسها قيمة الجانب الايسر من المعادلة

مثال :

```
Assigning the y value to the x variable          y = x;
```

لاحظ ان

العمليات الاتية تجمع بين كلاً من العمليات الحسابية وعمليات المساواة :

```
Adding x to the y variable                                  y += x;
Subtracting x from the y variable                           y -= x;
Multiplying the y variable by x                             y *= x;
Dividing the y variable by x                                y /= x;
Module x value of y                                         y %= x;
Logical shift of y representation to the right by x bit     y >>= x;
Logical shift of y representation to the left by x bit      y <<= x;
Bitwise operation AND                                       y &= x;
Bitwise operation OR                                        y |= x;
Bitwise operation exclusive OR
of x and y binary notations                                 y ^= x;
```

لاحظ أن

يمكن وضع عملية واحدة فقط من عمليات المساواة فى التعبير الواحد ، ويتم تنفيذ تلك العمليات ذات الارقام الصحيحة فقط

عمليات التحويل المنطقى تستخدم القيم من الارقام الثنائية أقل من 5 X

واكبر من ذلك يتم رفضة ، لذا فالتحويل يكون داخل المدى من 0 وحتى 31 بت

عمليات المقارنة :

عمليات المقارنة هى عمليات تقوم بمقارنة قيمتين وتنتهى بنتيجة من اثنين لا ثالث لهما وهما

TRUE وتعنى التطابق أو التساوى وتمثل بأى قيمة غير الصفر

FALSE وتعنى الاختلاف وعدم التساوى .. وتمثل بالقيمة صفر

0	(FALSE)
1,2,...	(TRUE)

مثال :

```
True if a equals b                          a == b;
True if a does not equal b                  a != b;
True if a is less than b                    a < b;
True if a is greater than b                 a > b;
True if a is less than or equals b          a <= b;
True if a is greater than or equals b       a >= b;
```

العمليات المنطقية :

يجب ان تكون مكتوبة فقط بصيغة حسابية (رياضية)

النتيجة تكون TRUE 1 اذا كانت قيمة المعامل تساوى FALSE 0

واذا كانت قيمة المعامل تختلف عن FALSE (0) تكون النتيجة FALSE (0)

مثال :

```
if(!a) Print("not 'a'");
```

العملية المنطقية OR (||) لقيم x و y .. تكون قيمة التعبير TRUE (1)

اذا كانت قيمة x او y صحيحه

والا تكون FALSE (0)

يتم حساب التعبيرات المنطقية هذه بالكامل ولا تطبق هنا طريقة الاستدعاء الجزئى

مثال :

```
if(x<0 || x>=max_bars) Print("out of range");
```

العملية المنطقية AND (&&) لقيمة x و y .. تكون قيمة التعبير TRUE (1)

اذا كانت قيمة كلاً من x و y صحيحه

وإلا تكون القيمة FALSE (0).

مثال :

```
if(p!=x && p>y) Print("TRUE");
```

العمليات الأخرى :

الفهرسة

وستخدم عند للوصول للعنصر الاول الى الرابع من المصفوفة وقيمة التعبير هى نفسها قيمة المتغير برقم تسلسلى للرابع

مثال :

```
array[i] = 3; //Assign the value of 3 to the i-th element of the array.
```

يمكن فقط للعدد الصحيح ان يكون فهرس للمصفوفة ، ويسمح فقط للمصفوفات الرباعية فما اقل وتتم فهرسة كل قياس من **0** الى قياس **1**

هناك حالة خاصة عند استخدام مجموعة ذات بـُعد واحد والتى تتكون من **50** عنصر

فالاشارة للعنصر الاول ستمثل كأنها مصفوفة فردية [**0**]

الى ان العنصر الاخير سيكون ممثل بـ [**49**]

للوصول الى ما بعد المصفوفة ، ولتنفيذ نظام فرعى سوف يؤدى لحدوث خطا اثناء التنفيذ

ERR_ARRAY_INDEX_OUT_OF_RANGE

والذى نشأ من الدالة **GetLastError**

عملية الفاصلة :

التعبيرات المفصول بينها بعلامة الفاصلة (،) يتم تنفيذها من الشمال الى اليمين

كل الحسابات الجانبية الخاصة بالتعبير اليسارى قد تظهر قبل حتى الشروع فى الشق الايمن (التعبير الايمن)

تتطابق النتيجة والقيمة مع التعبير الايمن

مثال :

```
for(i=0,j=99; i<100; i++,j--) Print(array[i][j]);
```

القواعد الأسبقية :

كل عملية فى هذا الجدول لها نفس الاولوية والاهمية

لاحظ ان العمليات المذكورة اولاً لها أولويو وأهمية اكبر من المذكورة أخراً

وهكذا

وظيفة القواعد الاسبقية هى تنظيم وتحديد العمليات والمعاملات

()	Function call	From left to right
[]	Referencing to an array element	
!	Logical negation	From right to left
-	Sign changing operation	
++	Increment	
--	Decrement	
~	Bitwise negation (complement)	
&	Bitwise operation AND	From left to right
|	Bitwise operation OR	
^	Bitwise operation exclusive OR	
<<	Left shift	
>>	Right shift	
*	Multiplication	From left to right
/	Division	
%	Module division	

+	Addition	From left to right
-	Subtraction	
<	Less than	From left to right
<=	Less than or equal	
>	Greater than	
>=	Greater than or equal	
==	Equal	
!=	Not equal	
||	Logical OR	From left to right
&&	Logical AND	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

ويتم تطبيق الاقواس التى لها اولوية اعلى لتغيير ترتيب تنفيذ العمليات

انتبة :

الأولوية لتنفيذ عمليات في MQL4 تختلف إلى حد ما من تلك في لغةC

..

يُتَبَع ،