د- العمليات

العمليات فى هذة اللغة هى وصف لبعض الخطوات الخوارزمية التى يجب تنفيذها لانجاز خطوة معينة ، اساس هذة اللغة هى مجموعة عمليات من هذا القبيل ويتم الفصل بينهم واحدة تلو الاخر بعلامة فاصلة منقوطة

يمكن للعملية هذة ان تشغل سطر واحد أو عدة اسطر فى البرنامج ، ويمكن وضع عملية او اكثر فى نفس البرنامج

العمليات التى تتداخل وتشترك فى وظائف معينة مثل (if, if-else, switch, while and for) يمكن ان تتداخل سوياً

مثال :

```
if(Month() == 12)
  if(Day() == 31) Print("Happy New Year!");
```

وتنقسم العمليات الى :

* العملية المركبة
* العملية التعبيرية
* عملية الكسر (جملة الكسر)
* عملية الاستمرار (جملة الاستمرار)
* عملية الرجوع (جملة الرجوع)
* العملية الظرفية if-else
* عملية التبديل
* عملية while
* عملية for

العملية المركبة :

العملية المركبة تتكون من عملية او اكثر من اى نوع وتكون محاطة بقوسين {}
بشرط الا تتبعها العلامة (;)

مثال :

```
if(x==0)
  {
   Print("invalid position x=",x);
   return;
  }
```

العملية التعبيرية :

أى تعبير يكون متبوعاً بهذة العلامة (;) تعتبر عملية .

واليك بعض الامثلة للعمليات التعبيرية :

عملية المهمات :

```
Identifier=expression;
x=3;
y=x=3; // error
```

عملية استدعاء الدوال :

```
Function_name(argument1,..., argumentN);

  FileClose(file);
```

يوجد أيضاً ما نسمية العملية الفارغة والتى تتكون من العلامة (;) فقط

وتستخدم لتشير الى وظيفة فارغة فى العملية

عملية الكسر(جملة الكسر) :

وهو العملية التى تكمن وظيفتها فى انهاء تنفيذ أقرب تحويل متداخل (وظيفة مشتركة)

إحدى أغراض هذا العملية هى الانتهاء من تنفيذ الدوال المتسلسلة عندما يتم تحديد قيمة معينة لمتغير

مثال :

```
// searching for the first zero element
for(i=0;i<array_size;i++)
  if((array[i]==0)
    break;
```

عملية الاستمرار (جملة الاستمرار) :

تعطينا عملية الاستمرار القدرة على التحكم فى بداية أقرب دالة خارجية

سواء كانت **while** او **for**

هذا العملية عكس تماماً وظيفة عملية الكسر

مثال :

```
// summary of nonzero elements of array
int func(int array[])
  {
   int array_size=ArraySize(array);
   int sum=0;
   for(int i=0;i<array_size; i++)
     {
      if(a[i]==0) continue;
      sum+=a[i];
     }
   return(sum);
  }
```

عملية الرجوع (جملة الرجوع) :

وظيفة عملية الرجوع هى انهاء تنفيذ الدالة الحالية وإعادة التحكم الى دوال الاستدعاء

تعبير الرجوع (أو عملية الرجوع) تقوم بهذة الوظيفة مع نقل النتيجة النهائية التى أُنتجت

يجب وضع هذة العملية بين قوسين وألا تحتوى على عملية اسناد (assignment operator)

مثال :

```
int CalcSum(int x, int y)
  {
   return(x+y);
  }
```

فى حالة الدوال ذات القيمة الفارغة عند الحاجة لرجوعها ؛ يجب الا تحتوى عملية الرجوع هنا على تعبيرات

مثال :

```
void SomeFunction()
  {
   Print("Hello!");
   return;    // this operator can be deleted
  }
```

العملية الظرفية if-else :

اذا كان التعبير صحيح ، يتم تنفيذ العملية رقم ١ ويتم اعطاء التحكم للعملية التى تلى وتتبع العملية رقم ٢

واذا كان التعبير خاطئ ، يتم تنفيذ العملية رقم ٢

```
if (expression)
   operator1
else
   operator2
```

لاحظ ان جزء **else** فى عملية **if** يمكن إهمالة هنا ، وبالتالى سيظهر الاختلاف عند تداخل الجزء المهمل مع عملية **if**

فى هذة الحالة ستوصل **else** الى اقرب عملية **if** سابقة فى نفس الجزء بشرط الا يحتوى على جزء **else**

مثال :

```
//   The else part refers to the second if operator:
  if(x>1)
  if(y==2) z=5;
  else     z=6;


//   The else part refers to the first if operator:
if(x>1)
  {
   if(y==2) z=5;
  }
else       z=6;


//   Nested operators
if(x=='a')
  {
   y=1;
  }
else if(x=='b')
  {
   y=2;
   z=3;
  }
else if(x=='c')
  {
   y = 4;
  }
else Print("ERROR");
```

عملية التبديل :

وتتم فيها المقارنة بين قيمة التعبير مع الثوابت فى كل المراحل وتعطى التحكم للعملية التى تتوافق مع قيمة هذا التعبير

كل متغير يمكن أن يتم تمييزة بثابت رقمى او حرفى ، ويجب ان يكون من النوع الصحيح

مثال :

```
switch(expression)
 {
  case constant: operators
  case constant: operators
    ...
  default: operators
 }
```

العمليات المرتبطة بالعنوان الرئيسى يتم تنفيذها اذا لم يتوافق الناتج مع احد ثوابت التعبيرات

المتغيرات الافتراضية يجب ان تكون محددة بنهاية ، اذا كان اى من الثوابت يتوافق مع قيمة التعبير والمتغير الافتراضى ، ، لا يتم تنفيذ اى شيئ

يتم حساب التعبير الثابت خلال التجميع ولا يمكن لثابتين فى نفس عملية التحويل ان تكون لهما نفس القيمة

مثال :

```
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;
  }
```

عملية while :

إذا كان التعبير صحيح ،يتم تنفيذ العملية حتى يصبح التعبير خطأ

وإذا كان التعبير خطأ ، سيتم تسليم التحكم للعملية التالية

```
while(expression)
 operator;
```

واذا كان هذا التعبير قد تم تعريفة مسبقاً ، وإذاً كان التعبير خطأ **false** من البداية فلن يتم تنفيذ العملية ابداً

مثال :

```
while(k<n)
  {
   y=y*x;
   k++;
  }
```

عملية for :

التعبير الأول **Expression1** يمثل بداية الحلقة ، والتعبير الثانى يمثل الاختبار التجريبى لنهاية الحلقة

اذا كان صحيحاً **true** سيتم تنفيذ المحتوى المطلوب للحلقة وسيتم اعادة المحاولة والتكرار الى ان تصبح قيمتة خطأ **false**

واذا كان خطأ **false** سيتم انهاء الحلقة وسيتم تسليم التحكم للعملية التالية

```
for (Expression1; Expression2; Expression3)
  operator;
```

```
Expression1;
while(Expression2)
 {
  operator;
  Expression3;
 };
```

يمكن لأى تعبير من الثلاث تعبيرات او كلهم ان لا يكونوا موجودين داخل عملية for ولكن العلامة (;) والتى تفصل بينهم يجب الا تهمل او تحذف

واذا أُهمل التعبير الثانى Expression 2 يعتبر دائماُ قيمتة true والعملية for (;;) هى حلقة متصلة للعملية while

مثال :

```
for(x=1;x<=7;x++) Print(MathPower(x,2));

for(;;)
  {
   Print(MathPower(x,2));
   x++;
   if(x>10) break;
  }

for(i=0,j=n-1;i<n;i++,j--) a[i]=a[j];
```

يُتبَع