برمجة الكائنات

(Object-Oriented Programming)

الـــــدوال

الدوال المعرفة بواسطة المستخدم Programmer-defined Functions

الدوال تمكن المبرمج من تقسيم البرنامج إلى وحدات modules كل دالة في البرنامج تمثل وحدة قائمة بذاتها، ولذا نجد أن المتغيرات المعرفة في الدالة تكون متغيرات محلية (Local) ونعنى بذلك أن المتغيرات تكون معروفة فقط داخل الدالة.

أغلب الدوال تمتلك لائحة من الوسائط (Parameters) والتي هي أيضاً متغيرات محلية. هنالك عدة أسباب دعت إلى تقسيم البرنامج إلى دالات

١ /تساعد الدوال المخزنة في ذاكرة الحاسب على اختصار البرنامج إذ يكتفي باستدعائها باسمها فقط لتقوم بالعمل المطلوب.

٢ /تساعد البرامج المخزنة في ذاكرة الحاسب أو التي يكتبها المستخدم على تلافى عمليات التكرار في خطوات البرنامج التي تتطلب عملاً مشابها لعمل تلك الدوال.

٣ /تساعد الدوال الجاهزة في تسهيل عملية البرمجة.

٤ /يوفر استعمال الدوال من المساحات المستخدمة في الذاكرة.

كل البرامج التي رأيناها حتى الآن تحتوى على الدالة main وهي التي تنادى الدوال المكتبية لتنفيذ مهامها .سنرى الآن كيف يستطيع المبرمج بلغة ال سي ++ كتابة دوال خاصة به.

تعريف الدالة Function Definition

يأخذ تعريف الدوال الشكل العام التالي:

1- النوع الاول من كتابه الدوال .

وهذا النوع من الدوال لا يستقبل بيانات ولا يرجع اي قيمة .

```
function-name ( )
{
declarations and statements
}
```

حيث :

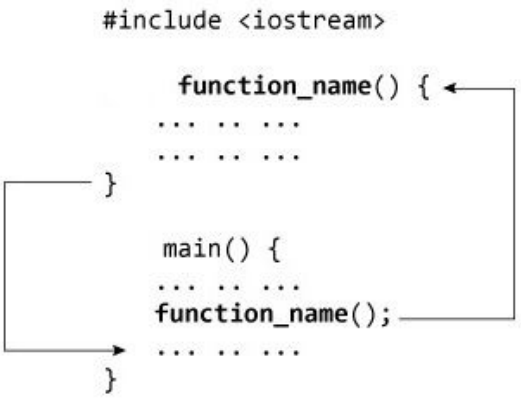
function-name:اسم الدالة والذي يتبع في تسميته قواعد تسمية المعرفات.

declarations and statements:

تمثل جسم الدالة والذي يطلق عليه في بعض الأحيان block يمكن أن يحتوى ال block على إعلانات المتغيرات ولكن تحت أي ظرف لا يمكن أن يتم تعريف دالة داخل جسم دالة أخرى .السطر الأول في تعريف الدالة يدعى المصرح declarator والذي يحدد اسم الدالة ونوع البيانات التي تعيدها الدالة وأسماء وأنواع وسيطاتها.

Ex.1

```
#include <iostream.h>
main ()
{
 print ();
}
Print ()
{
cout << "Welcome to OOP ";
}
```



الشكل يوضح كيف يتم استدعاء الداله الثانوية داخل البرنامج الرئيسي

Ex.2

```
#include <iostream.h>
addition ()
{
 int a,b,r;
 a=10;
 b=20;
 r=a+b;
 cout<<r;
}
main ()
{
 addition ();
}
```

Ex.3

```
#include <iostream.h>
addition ()
{
 int a,b,r;
 cout <<"Enter a And b";
 cin >> a >> b;
 r=a+b;
 cout<<r;
}
main ()
{
```

```
addition ();
}
```

2- النوع الثاني من كتابه الدوال .

وهذا النوع من الدوال يستقبل قيم ولا يرجع اي قيمة .

```
function-name (parameter list )
{
declarations and statements
}
```

حيث :

function-name:اسم الدالة والذي يتبع في تسميته قواعد تسمية المعرفات .

parameter list : هي لائحة الوسيطات الممرة إلى الدالة وهي يمكن أن تكون خالية (void) أو تحتوى على وسيطة واحدة أو عدة وسائط تفصل بينها فاصلة ويجب ذكر كل وسيطة على حدة.

declarations and statements: تمثل جسم الدالة والذي يطلق عليه في بعض الأحيان block يمكن أن يحتوى ال block على إعلانات المتغيرات ولكن تحت أي ظرف لا يمكن أن يتم تعريف دالة داخل جسم دالة أخرى .السطر الأول في تعريف الدالة يدعى المصرح declarator والذي يحدد اسم الدالة ونوع البيانات التي تعيدها الدالة وأسماء وأنواع وسيطاتها.

ex1.

مثال لضرب رقم يقراء من قبل المستخدم مع الرقم 2

```
#include <iostream>
addition (int a)
{
  int r;
  r=a*2;
cout << r;
```

```
}
 main ()
{
 addition (5);
}
```

ex2

مثال لجمع رقم يقراء من قبل المستخدم مع الرقم 4

```
#include <iostream>
addition (int a)
{
 int r;
 r=a+4;
 cout << r;
}
 main ()
{
 int x;
 cout << ""x";
 cin >> x;
 addition (x);
}
```

ex3

مثال الحاسبة يتكون من اربع دوال للعمليات الاربعة.

```
#include <iostream>
 main ()
```

```
{
 int x,y;
 cout << ""x,y";
 cin >> x >>y;
 addition (x,y);
 sub(x,y);
 Muilt(x,y);
 Div(x,y);

}


addition (int a,int b)
{
 int r;
 r=a+b;
cout <<"the sum = " << r;
}
sub (int a, int b)
{
 int r;
 r=a-b;
 cout <<"the sub = " << r;
}

Muilt (int a, int b)
```

```
{
 int r;
 r=a*b;
 cout <<"the Muilt = " << r;
}

Div (int a, int b)
{
 int r;
 r=a/b;
 cout <<"the Div = " << r;
}
```

```
# include <iostream>

main() {
... .. ...
      add(num1, num2);  // Actual parameters: num1 and num2
... .. ...
}

add(int a, int b) {     // Formal parameters: a and b
... .. ...
 add = a+b;
... .. ...
}
```

الشكل يوضح كيفية استدعاء القيم واستخدمها داخل الداله الثانوية

ex

داله معرفه الرقم الاكبر مابين ثلاث ارقام باستخدام داله ثانوية .

```
#include <iostream>
 main ()
```

```
{
int x,y,z;
cin >> x >> y>>z;
max(x,y,z);
}
max(int a,int b, int c)
{
  int m;
  m=a;
if((a>b)&&(a>c))
  {
    cout< "The a Is Max Number";
  }
if((b>a)&&(b>c))
  {
    cout< "The b Is Max Number";
  }
if((c>a)&&(c>b))
  {
    cout< "The c Is Max Number";
  }
}
```