كيفية كتابة برنامج ب C++

```
#include<iostream.h>
main ( )
{
cout <<''welcome to C++ !\n'';
return 0;
}
```

الخرج من البرنامج:

welcome to C++ !

يقوم الحاسوب بتنفيذ البرنامج ويعود سريعاً للمحرر

من الآن فصاعداً، إذا أردت تثبت المخرجات على الشاشة عليك إضافة التالي إلى البرنامج :

```
#include <conio.h>
getch( )
```

iostream.h:

وهو ملف يجب تضمينه مع أي برنامج يحتوى على عبارات تطبع بيانات على الشاشة أو تستقبل بيانات من لوحة المفاتيح. يسمى iostream ملف ترويسة ،(header file) وهنالك الكثير من ملفات الترويسة الأخرى، فمثلاً إذا كنا نستعمل في برنامجنا دالات رياضية ك ( )cosو( )sin نحتاج إلى شمل ملف ترويسة يدعى math.h

الدالة main :-

main( )

يبدأ تشغيل أي برنامج C++ من دالة تدعي main( )، وهي دالة مستقلة ينقل نظام التشغيل التحكم إليها وهي جزء أساسي في برنامج C++. الأقواس بعد main تشير إلى أن main هي عبارة عن دالة. قد يحتوى برنامجC++ على أكثر من دالة إحداهما بالضرورة هيmain( )

يبدأ تنفيذ البرنامج من الدالة main حتى لو لم تكن هي الأولى في سياق البرنامج يتم حصر جسم الدالة main بأقواس حاصرة{ }

إليك الآن مثالاً لبرنامج يستقبل رقمين من المستخدم ويجمعهما ويعرض ناتج الجمع-:

```
#include<iostream.h>
#include<conio.h>
main ( ) {
int integer1, integer2, sum;
cout <<"Enter first integer\n";
cin >> integer1;
cout <<"Enter second integer\n";
cin >> integer2;
sum= integer1+integer2;
cout <<"sum="<<sum<<endl;
getch();
return 0;
}
```

أنواع البيانات الأساسية:

الأنواع.

اسم النوع	يستعمل لتخزين	أمثلة عن القيم المخزنة
char	أحرف	''a''
short	أرقام صحيحة قصيرة	222
int	أرقام صحيحة عادية الحجم	153,406
long	أرقام صحيحة طويلة	123,456,789
float	أرقام حقيقية قصيرة	3,7
double	أرقام حقيقية مزدوجة	7,533,039,395
long double	أرقام حقيقية ضخمة	9,176,321,236,01202,6

العباره If

```
#include <iostream.h>
main ( )
{
int num1 , num2;
cout << " Enter two integers, and I will tell you\n"
<<" the relation ships they satisfy: ";
cin >> num1>> num2;
if (num1== num2)
cout << num1 << " is equal to " << num2 << endl;
if (num1!= num2)
cout << num1 << " is not equal to " << num2 << endl;
if (num1< num2)
```

```
cout << num1 << " is less than " << num2 << endl;
if (num1> num2)
cout << num1 << " is greater than " << num2 << endl;
if (num1<= num2)
cout << num1 << " is less than or equal to " << num2
<< endl;
if (num1>= num2)
cout << num1 << " is greater than or equal to " << num2
<< endl;
return 0;
}
```

العباره if - else

```
#include <iostream.h>
main ( )
{
int grade ;
cout << " Enter the grade";
cin >>grade;
if(grade>= 50)
cout<<"pass" <<endl;
else
cout <<"fail"<<endl;
```

```
return 0;
}
```

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 وتسمى هذه العملية

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:

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

حيث:

return-value-type: نوع القيمة المعادة بواسطة الدالة والذي يمكن أن يكون أي نوع من أنواع بيانات وإذا كانت الدالة لا ترجع أي قيمة يكون نوع اعادتها void

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

إستدعاء الدالة Function Call

يتم استدعاء الدالة التسبب بتنفيذها من جزء آخر من البرنامج، العبارة التي تفعل ذلك هي استدعاء الدالة (يؤدى استدعاء الدالة إلي انتقال التنفيذ إلي بداية الدالة .يمكن تمرير بعض الوسيطات إلى الدالة عند استدعائها وبعد تنفيذ الدالة يعود التنفيذ للعبارة التي تلي استدعاء الدالة.

قيم الإعادة Returned Values

بإمكان الدالة أن تعيد قيم إلي العبارة التي استدعتها .ويجب أن يسبق اسم الدالة في معرفها وإذا كانت الدالة لا تعيد شيئاً يجب استعمال الكلمة الأساسية void كنوع إعادة لها للإشارة إلي ذلك. هنالك ثلاث طرق إرجاعالتي يمكن بها التحكم إلى النقطة التي تم فيها استدعاء الدالة:

1. إذا كانت الدالة لا ترجع قيمة يرجع التحكم تلقائياً عند الوصول إلى نهاية الدالة.
2. باستخدام العبارة return;
3. إذا كانت الدالة ترجع قيمة فالعبارة return expression; تقوم بإرجاع قيمة التعبير expression إلى النقطة التي استدعتها.

خذ برنامجاً يستخدم دالة تدعى square لحساب مربعات الأعداد من 1 الى 10 .

```
#include<iostream.h>

int square(int);//function prototype

main()
{
for(int x=1;x<=10;x++)
cout<<square(x)<<" ";
cout<<endl;
}

//now function definition
int square(int y)
```

```
{
return y*y;
}
```

الخرج من البرنامج يكون كالآتي:

```
1 4 9 16 25 36 49 64 81 100
```

يتم استدعاء الدالة **square** داخل الدالة **main** وذلك بكتابة **square(x)**. تقوم الدالة **square** بنسخ قيمة **x** في الوسيط **y** . ثم تقوم بحساب **y*y** ويتم إرجـــاع النتيجة إلى الدالة **main** مكان استدعاء الدالة **square**، حيث يتم عرض النتيجة وتتكرر هذه العملية عشر مرات باستخدام حلقة التكرار **for** .

تعريف الدالة **square( )** يدل على أنها تتوقع وسيطة من النـــوع **int** . و **int** التي تسبق اسم الدالة تدل على أن القيمة المعادة من الدالة **square** هي من النـــوع **int** أيضاً . العبارة **return** تقوم بإرجاع ناتج الدالة إلى الدالة **main**.

السطر:

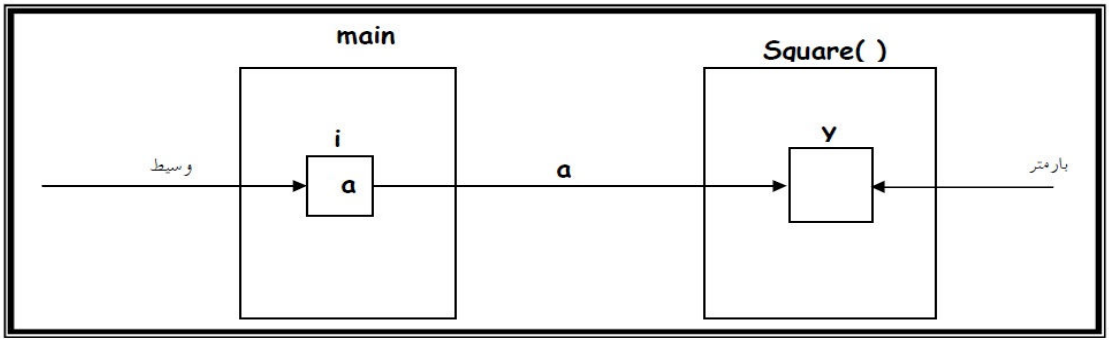
```
int square (int)
```

هو نموذج أو تصريح الدالة **(function prototype)** .

الوسيطات Parameters

الوسيطات هي الآلية المستخدمة لتمرير المعلومات من استدعاء الدالة إلى الدالة نفسها حيث يتم نسخ البيانات وتخزن القيم في متغيرات منفصلة في الدالة تتم تسمية هـــذه المتغيرات في تعريف الدالة.فمثلاً في المثال السابق تؤدى العبارة `cout<< square(a);` في ( )main إلى نسخ القيمة a إلى البارمتر y المعرف في تعريف الدالة.

المصطلح وسيطات Argument يعنى القيم المحددة في استدعاء الدالة بينما يعنى المصطلح بارمترات parameters المتغيرات في تعريف الدالة والتي تم نسخ تلـــك القيم إليها، ولكن غالباً ما يتم استعمال المصطلح وسيطات لقصد المعنيين.



البرنامج التالي يستخدم دالة تدعى maximum والتي نرجع العدد الأكبر بين ثلاثة أعداد صحيحة.

يتم تمرير الأعداد كوسائط للدالة التي تحدد الأكبر بينها وترجعه للدالة main باستخدام العبارة return ويتم تعيين القيمة التي تمت إعادتها إلى المتغير largest الذي تتم طباعته.

```
#include <iostream.h>
int maximum (int, int, int);
main( )
{
```

```
int a, b, c;
cout << "Enter three integers: " ;
cin >> a >> b >> c ;
cout << " maximum is : " << maximum (a, b, c) << endl;
return 0;
}
int maximum (int x, int y, int z)
{
int max = x;
if (y > x)
max = y;
if (z > max)
max = z;
//Continued
return max;
}
```