الدرس الثالث: الدوال

Functions

مقدمة

الدوال أو الروتينات الفرعية عبارة عن برامج صغيرة أو مجموعة من التعليمات تقوم بأداء مهمة معينة. يخصص لهذا البرنامج اسم ويقع استدعاؤه داخل الدالة الرئيسية ()main أو في أي دالة أخرى.

لاستخدام الدوال عدة فوائد نذكر منها:

- عدم تكرار التعليمات داخل البرنامج حيث يتم إنشاء دالة واحدة ويقع استدعاؤها كلما دعت الحاجة لذلك
- جعل الدالة الرئيسية للبرنامج أقل تعقيدا وغير مزدحمة بالأوامر
- تقسيم البرنامج إلى أجزاء يمكن اختبارها منفصلة مما يمكّن من تحديد الأخطاء بسرعة
- توفير المجهود والوقت والتفكير عبر إنشاء مكتبة خاصة تمكّن من إعادة استخدام الدوال في أي برنامج
- تقسيم العمل بين المبرمجين عند العمل في المشاريع الجماعية الكبيرة
- تبادل الخبرات بين مطوري البرامج بنشر أجزاء يستخدمها الآخرون في برامجهم.

1- الدوال القياسية المعرفة داخل لغة C++

الدوال القياسيةعبارة عن صيغ معرفة مسبقا وتستدعى داخل البرامج لإنجاز عمل ما . إذا كانت هذه الدوال ترجع قيمة فيجب أن تعيّن أو تسند إلى متغير من نفس نوع القيمة المسترجعة.

تنقسم الدوال القياسية إلى عدة أنواع: دوال الرياضيات، دوال السلاسل الحرفية، دوالالتحويل، دوال الإدخال والإخراج ، دوال الرسوم البيانية ...الخ.

على سبيل المثال، يبين الجدول التالي بعض النماذج من الدوال الرياضية وهي كما ذكرنا مجمعة داخل الملف الرأسي <cmath>. في هذا الجدول نفترض أن x و y متغيرين حقيقيين من نوع double :

الدالة	الوصف	أمثلة
fabs(x)	حساب الجذر التربيعي للعدد x	fabs(3.5) = 3.5 fabs(-3.5) = 3.5
sqrt(x)	حساب الجذر التربيعي للعدد x	sqrt(16.0) = 4
pow(x,y)	ترجع قيمة x مرفوعة للأس y أي x^y	pow(2,3) = 8 pow(9,0.5) = 3
fmod(x,y)	حساب باقي قسمة x على y	fmod(8, 3) = 2
cos(x)	حساب جيب التمام لمعطى الدالة	cos(0) = 1 cos(3.14) = -1
sin(x)	حساب جيب الزاوية لمعطى الدالة	sin(0) = 0 sin (1.57) = 1
tan(x)	حساب ظل الزاوية لمعطى الدالة	tan(0) = 0
log(x)	حساب اللوغاريتم الطبيعي لمعطى الدالة	log(1) = 0 log(2.71828) = 1

exp(x)	ترجع (e ≈ 2.71828) أس x أي e^x	exp(0) = 1 exp(1) = 2.71828
floor(x)	التقريب لأقرب وأصغر عدد صحيح	floor(2.3) = 2 floor(-3.5) = -4
ceil(x)	التقريب لأقرب وأكبر عدد صحيح	ceil(2.3) = 3 ceil(-3.5) = -3

2- الدوال المعرفة من طرف المستخدم

يمكن للمستخدم تعريف دوال جديدة باستخدام الصيغة التالية:

```
Return_Type  Function_Name (Parameter_List)
{
      // (Function_Body) مجموعة من التعليمات
      return  القيمة المسترجعة;
}
```

حيث تتكون أي دالة من العناصر التالية:

- نوع البيانات التي تقوم الدالة بإرجاعها (Return_Type). أما إذا كانت الدالة لا تقوم بإرجاع أي بيانات فتكون من النوع **void**.
- إسم الدالة (Function_Name) وهو المعرف المستخدم لاستدعاء الدالة.
- قائمة بمعاملات أو وسائط الدالة (Parameter_List) وتسمى أيضا متغيرات التخاطب وهي القيم التي يتم تمريرها للدالة في كل مرة يتم استدعاؤها.
- جسم الدالة (Function_Body) ويحتوي على الكود الذي يتم تنفيذه عند استدعاء الدالة.

<u>مثال رقم 1</u>

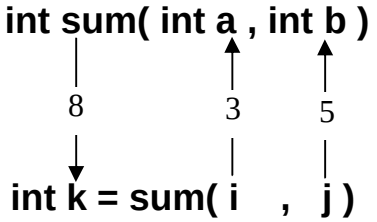
```
#include <iostream>
using namespace std;

int sum(int a, int b)          // sum تعريف الدالة
{
      return a+b;
}

int main( )
{
      int i = 3,  j = 5;
      int k = sum(i,j);        // sum  استدعاء الدالة
      cout<<"Sum = "<<k;
      cout<<endl;
      return 0;
}
```

- تقوم الدالة sum بإرجاع مجموع عددين صحيحين.

- في هذا البرنامج يتم التخاطب بين الدالة الرئيسية main والدالة sum عن طريق المعاملات وذلك على النحو التالي:



مثال رقم 2: أكتب دالة اسمها sqr تقوم بحساب مربع عدد حقيقي x.

الحل

```
float sqr(float x)
{
      return x*x;
}
```

لاستدعاء هذه الدالة داخل الدالة الرئيسية نكتب مثلا:

```
float y = 1.5 + sqr(3);
```

مثال رقم 3

أكتب دالة اسمها min تقوم بإرجاع القيمة الأصغر من بين العددين الصحيحين a و b :

الحل

```
int min(int  a,  int  b)
{
      if (a <= b)
            return a;
      else
            return b;
}
```

ملاحظة: عند استدعاء أي دالة يجب احترام **عدد** و**نوع** و **ترتيب** المعاملات التي يقع تمريرها للدالة.

على سبيل المثال، يحتوي الجدول التالي على عمليات استدعاء صحيحة للدالة min وأخرى خاطئة نظرا لعدم احترامها للقاعدة المذكورة:

عمليات استدعاء مقبولة	عمليات استدعاء غير مقبولة

m = min(3,5);
m = min(min(i,j),k);
m = min(min(i,j),min(k,l));
	m = min(3.5,5);
	m = min(i,j,k);
	m = min(i,j,k,l);

مثال رقم 4

أكتب دالة اسمها maximum تقوم بإرجاع القيمة الكبرى من بين ثلاث أعداد حقيقية x، y و z.

الحل

```
float maximum (float  x,   float  y,   float  z)
{
      float max = x;
      if (y > max)
            max = y;
      if (z > max)
            max = z;
      return max;
}
```

3- المتغيرات العامة والمتغيرات المحلية

المتغيرات العامة (Global Variables) هي متغيرات يعلن عنها خارج الدالة الرئيسة للبرنامج وتكون متاحة لكل الدوال لاستخدامها دون إعادة الاعلان عنها.

أما المتغيرات المحلية (Local Variables) فيعلن عنها داخل دالة معيّنةلا يمكن استخدامها الا داخل الدالة نفسها ، وبانتهاء عمل الدالة ينتهي مفعولها وتفقد قيمتها.

مثال

```
#include <iostream>
using namespace std;

int x, y, z;            // متغيرات عمومية

int Sum(int a, int b)
{
      int c;            // متغير محلي
      c = a + b;
      return c;
}

int main()
{
      x = 7;
      y = 8;
      z = Sum(x,y);
      cout<<"z = "<<z<<endl;
      return 0;
}
```

ملاحظة: يفضل استخدام المتغيرات المحلية عن المتغيرات العمومية لما قد ينتج عن ذلك من آثار جانبية إذ أن كل تغيير يمس المتغيرات العمومية يؤثر على قيمتها في كل أنحاء البرنامج.

4- الاستدعاء بالقيمة والاستدعاء بالعنوان (Call by value and call by reference)

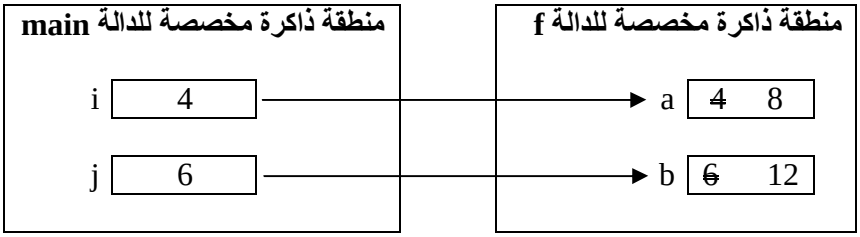
لنفترض البرنامج التالي :

```
#include <iostream>
using namespace std;

void f(int a, int b)
{
	a = 2 * a;
	b = 2 * b;
	cout<<"a = "<<a<<"\t"<<"b = "<<b;
	cout<<endl;
}

int main()
{
	int i = 4;
	int j = 6;
	f(i,j);		// استدعاء الدالة f
	cout<<"i = "<<i<<"\t"<<"j = "<<j;
	cout<<endl;
	return 0;
}
```

- تسمى المعاملات a و b المستخدمة في الدالة f **معاملات صورية** (Virtual Parameters).
- تسمى المتغيرات i و j المستخدمة عند استدعاء الدالة **معاملات حقيقية** (Actual Parameters).
- تستخدم الدالة f **طريقة الاستدعاء بالقيمة** (Call bay value) حيث يتم الحصول على نسخ من متغيرات التخاطب الحقيقية والتعامل معها داخل الدالة وبالتالي فإن التغيرات التي تطرأ على المعاملات الصورية لن تؤثر على قيمة المعاملات الحقيقية (شكل 6).



شكل 6: تمرير المعاملات بالقيمة

وبالتالي فإن نتيجة تنفيذ البرنامج تكون كما يلي:

```
a = 8   b = 12
i = 4   j = 6
Press any key to continue . . .
```

على العكس من ذلك، في **طريقة الاستدعاء بالعنوان** (Call by reference) تتعامل الدالة مع النسخ الأصلية لمتغيرات التخاطب مما يجعل كل تغيير يطرأ على المعاملات الصورية ينعكس مباشرة على المعاملات الحقيقية.

لاستخدام هذه الطريقة، يكفي تغيير واجهة الدالة لتصبح:

void f(int &a, int &b);

يسمى الرمز **&** معامل العنوان. وفي هذه الحالة تكون نتيجة البرنامج كما يلي:

```
a = 8    b = 12
i = 8    j = 12
Press any key to continue . . .
```

ملاحظة

إن الاستخدام المفرط لطريقة الاستدعاء بالعنوان قد يؤدي إلى تأثيرات جانبية خطيرة كتغيير قيم لا يرغب المبرمج في تغييرها، لذلك ينصح باستخدام هذه الطريقة فقط عند الضرورة (إذا كانت الدالة مطالبة بإرجاع أكثر من قيمة مثلا).

مثال : أكتب دالة اسمها circle تقوم بإرجاع محيط ومساحة دائرة معينة من خلال معرفة نصف القطر.

الحل

```
void circle(float radius, float &perimeter, float &area)
{
      const float PI = 3.14;
      perimeter = 2*PI*radius;
      area = PI*radius*radius;
}
```

في هذه الدالة وقع استخدام طريقة الإستدعاء بالقيمة مع المدخل radius وطريقة الإستدعاء بالعنوان مع المخرجات perimeter و area.

5- الدوال المحمّلة Overloaded Functions

الدوال المحملة هي دوال لها نفس الاسم ولكنها تنجز عمليات مختلفة معتمدة على عدد ونوع البيانات التي يقع تمريرها عند الاستدعاء.

في لغة C++ لا يسمح المترجم بالإعلان عن متغيرين لهما نفس الاسم في نفس النطاق. كذلك لا يمكن أن نصرح بدالتين لهما نفس الاسم إلا إذا كانتا مختلفتين من حيث عدد أو نوع المعاملات.

مثال رقم 1

```
#include <iostream>
using namespace std;

int square(int i)
{
      return i*i;
}

float square(float f)
{
      return f*f;
}

int main( )
{
      int a = 3;
      float x = 2.5;
      cout<<"The square of a is "<<square(a); // إستدعاء الدالة الأولى
```

```
        cout<<endl;
        cout<<"The square of x is "<<square(x); // إستدعاء الدالة الثانية
        cout<<endl;
        return 0;
}
```

في هذا البرنامج صرحنا بدالتين لهما نفس الاسم square لكن لا تتعاملان مع نفس أنواع البيانات فالأولى تقوم بحساب مربع عدد صحيح (int) والثانية تقوم بحساب مربع عدد حقيقي (float).

وبالتالي فإن نتيجة تنفيذ البرنامج تكون كما يلي:

```
The square of a is 9
The square of x is 6.25
Press any key to continue . . .
```

مثال رقم 2

أكتب برنامجا بلغة C++ يحتوي على دالتين بنفس الاسم Area تقوم الأولى بحساب مساحة دائرة وتقوم الثانية بحساب مساحة مستطيل.

الحل

```
#include <iostream>
using namespace std;

float Area (float);          // حساب مساحة دائرة من خلال معرفة نصف القطر
float Area (float, float); // حساب مساحة مستطيل من خلال معرفة الطول والعرض

int main( )
{
      float R;    // نصف قطر الدائرة
      float L, W; // طول وعرض المستطيل
      cout<<"Enter the radius of the circle (in cm): ";     cin>>R;
      cout<<"The area of the circle is "<<Area(R)<<" cm2";
      cout<<endl;
      cout<<"Enter the length of the rectangle (in cm): ";  cin>>L;
      cout<<"Enter the width of the rectangle (in cm): ";   cin>>W;
      cout<<"The area of the rectangle is "<<Area(L,W)<<" cm2";
      cout<<endl;
      return 0;
}

float Area(float Radius)
{
      const float PI = 3.14;
      return PI * Radius * Radius;
}

float Area (float Length, float Width)
{
      return Length*Width;
}
```

تتبع تنفيذ البرنامج

```
Enter the radius of the circle (in cm): 10
The area of the circle is 314 cm2

Enter the length of the rectangle (in cm): 4
Enter the width of the rectangle (in cm): 3.5
The area of the rectangle is 14 cm2

Press any key to continue . . .
```

ملاحظة: في البرنامج السابق وقع تعريف الدوال Area() بعد الدالة الرئيسية main() لذلك وجب الإعلان عن هاتين الدالتين في بداية البرنامج.

6- الدوال الضمنية أو السطرية Inline Functions

الدالة الضمنية هي دالة يتم إدراج جسمها في كل نقطة في البرنامج يتم استدعاؤها فيها.

لإنشاء دالة ضمنية يجب علينا أن نسبق تعريف هذه الدالة بالكلمة المفتاحية **inline** كما هو موضح في المثال التالي:

```
inline void exchange(int &x, int &y)
{
      int temp = x;
      x = y;
      y = temp;
}
```

ملاحظات

1- لا تتناسب هذه الطريقة إلا مع الدوال الصغيرة جدا التي لا يؤدي إدراجها إلى تضخيم حجم البرنامج.
2- لتجنب إعادة تعريف الدالة الضمنية في كل ملف ستستخدم فيه، يحبذ تعريفها في ملف رأسي (Header File).

تمرين

أكتب دالة إسمها DaysPerMonth تقوم بإرجاع عدد الأيام في شهر ميلادي معين.

الشهر	1	2	3	4	5	6	7	8	9	10	11	12
عدد الأيام	31	28/29	31	30	31	30	31	31	30	31	30	31

السنة الكبيسة (leap year) هي سنة تحتوى على 366 يوما حيث يضاف يوم إلى شهر فبراير في السنة الكبيسة ليصبح متضمنا على 29 يوم بدلا من 28.

قاعدة عامة

- تعتبر السنوات التي تقبل القسمة على 100 كبيسة فقط إذا كانت تقبل القسمة على 400.
على سبيل المثال، السنوات 2000 و 2400 كبيسة، أما السنة 2100 فهي ليست كبيسة.

- بقية السنوات تعتبر كبيسة فقط إذا كانت تقبل القسمة على 4.
على سبيل المثال، السنوات 1996 و 2004 كبيسة.

الحل

```
int  DaysPerMonth(int month, int year)
{
      int days = 31;
      if ((month==4)||(month==6)||(month==9)||(month==11))
            days = 30;
      if (month = = 2)
      {
            if (((year%100!=0)&&(year%4==0))||(year%400==0))
                  days = 29;
            else
                  days = 28;
      }
      return days;
}
```

7- الماكرو Macros

الماكرو مجموعة بسيطة من التعليمات تقوم بإرجاع نتيجة معينة. يتم إنشاء الماكرو مرة واحدة ثم يقع استدعاؤها كلما احتجنا إليها.

الصيغة العامة

```
#define  macro_name  instruction
```

مثال

```
#include <iostream>

#define sum(a,b) a+b

#define mul(a,b) a*b

#define max(a,b) ((a >= b)? a : b)

using namespace std;

int main()
{
      int i, j, min;
      cout<<"Enter the first value : ";
      cin>>i;
      cout<<"Enter the second value : ";
      cin>>j;
      cout <<"Sum = "<<sum(i,j);
      cout <<"Product = "<<mul(i,j);
      cout <<"Max = "<<max(i,j);
      cout<<endl;
      return 0;
```

```
}
```

تتبع تنفيذ البرنامج

```
Enter the first value : 6
Enter the second value : 4

Sum = 10
Product = 24
Max = 6

Press any key to continue . . .
```

تمرين

1- قم بإنشاء ملف رأسي يحمل إسم "Circle.h" ويحتوي على دالتين:

- الدالة float area(float radius) التي تقوم بحساب مساحة دائرة معينة باستخدام الصيغة

$$Area = 3.14 \times radius^2$$

- الدالة float perimeter(float radius) التي تقوم بحساب محيط دائرة معينة باستخدام الصيغة

$$Perimeter = 2 \times 3.14 \times radius$$

2- قم بإنشاء ملف مصدر يحمل إسم "Circle.cpp" يقوم بقراءة نصف قطر دائرة معينة ثم يحسب المساحة والمحيط باستخدام دوال الملف الرأسي "Circle.h".

- **الحل**

```
// circle.h
const float PI = 3.14;

float area(float radius)
{
   return PI*radius*radius;
}

float perimeter(float radius)
{
   return 2*PI*radius;
}
```

```
// circle.cpp
#include <iostream>
#include "circle.h"     // إدراج الملف الرأسي
using namespace std;

int main( )
{
   float r;
   cout<<"Enter the radius (in cm) : ";
   cin>>r;
   cout<<"Area = "<<area(r)<<" cm2"<<endl;
   cout<<"Perimeter = "<<perimeter(r)<<" cm";
   cout<<endl;
   return 0;
}
```

- **تتبع تنفيذ البرنامج**

```
Enter the radius (in cm) : 10

Area = 314 cm2
Perimeter = 62.8 cm

Press any key to continue . . .
```