C++ Programming

First Stage

By

Dr. Hussam Jasim Mohammed

الفصل
1

مقدمة عن البرمجة
Introduction

البرمجة (Programming):
عبارة عن مجموعة من الأسس والمبادئ والنظريات التي تتلائم مع البيئات العملية "مبادئ علم الحاسوب".

البرنامج (Program):
- تعريف عام : مجموعة من التعليمات "Codes" والأوامر المرتبة لحل مشكلة معينة.
- أو : مجموعة من الأكواد المكتوبة بلغة من لغات البرمجة.

الحزم (Package):
مجموعة من البرامج الجاهزة المتكاملة والمترابطة فيما بينها، والتي تؤدي وظائف متعددة وتعمل تحت بيئة واحدة.
برامج جاهزة: يعني لا يتم التعديل عليها فهي صيغة نهائية مثل : windows, office package

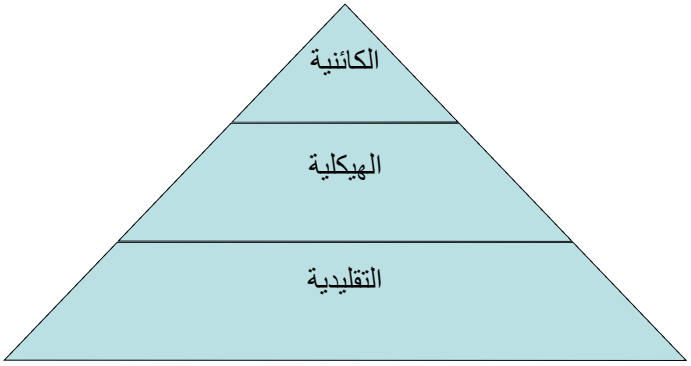
أنواع البرامج Program types:
١- أنظمة التشغيل (OS).
٢- لغات البرمجة (Programming Languages).
٣- التطبيقات (Applications).
٤- المفسرات/المترجمات (Compilers).

مستويات البرمجة:
وهي مستويات يمر عليها المبرمج خلال رحلته التعليمية البرمجية:

١) التقليدية: وهي مرحلة كتابة برامجه بطريقة عشوائية لا تعتمد على أسس ونظريات وهيكلة، فما يهم المبرمج هو الوصول لحل المشكلة فقط.
عيوبها:
- لا يمكن اكتشاف الخطأ في البرنامج.
- لا يمكن تطوير البرنامج بسهولة.
- تطوير البرنامج يزيد حجم البرنامج بشكل كبير.

٢) الهيكلة: وهي مرحلة يقوم فيها المبرمج بتقسيم برنامجه إلى هياكل تساعد على اكتشاف الأخطاء والتطوير كما أعطت إمكانيات كثيرة.

٣) الكائنية: وهي مرحلة يقوم فيها المبرمج بهيكلة برنامجه إلى كائنات كل كائن يتخصص في حل مشكلات في مجال معين، ويحتوي الكائن على طرق "دوال" مترابطة تجعل من الكائن يمتلك ذكاء في تحديد الاستجابة المطلوبة تلقائياً.



مستويات لغات البرمجة (Programming Language Levels):

١. **المستوى الأدنى (Lowest Language Level (L.L.L:**

يتعامل هذا المستوى مع الدوائر المنطقية ويتكون من :

- لغة الآلة (Assembler).
- لغة التجميع (Micro Assembly).

عيوبه:

- صعوبة كتابة البرمجيات أو فهمها لأنها تتعامل مع رموز بالنظام الثنائي "0/1".
- لغة قريبة من الآلة وبعيدة عن الإنسان.
- تحتاج إلى متخصصين في الحاسوب.

ملاحظة:

نظام الإدخال (ثنائي)
نظام الحفظ (عشري).
نظام العرض (سادس عشر).

٢. **المستوى المتوسط (Midst Language Level (M.L.L:**

ظهر هذا المستوى ثم اختفى بسرعة بسبب سرعة التطوير إلى المستويات العليا حيث دمجت وصنفت لغاته في المستوى العالي ويتكون من اللغات التالية:

- C.
- C++.

مميزاته:

- قريبة من الحاسوب ومن الإنسان.

٣. **المستوى العالي (Highest Language Level (H.L.L:**

تتكون من :

- Basic.
- Pascal (تستخدم في المجال العلمي).
- Fortran (تستخدم في المجال الفيزيائي والرياضي).
- Cobol (تستخدم في المجال التجاري).
- Java.

مميزاته:

- قريبة جداً من لغة الإنسان.
- سهلة الفهم والكتابة.
- لغات هيكلية.
- لغات متخصصة (كل لغة تهتم بجانب معين، وبالتالي يمكن الاستفادة من اللغات المختلفة بحسب نوع المشكلة المراد حلها).

الفصل
2

مقدمة عن C++
Introduction C++

نبذة تاريخية:
أول لغة ظهرت هي الأسمبلي للتعامل مع البوابات المنطقية ومبادئ الحوسبة، ثم ظهرت مبادئ لغة C في الأربعينيات.

مميزات لغة C :

- لغة كاملة وشاملة (تعتبر أم لغات البرمجة).
- بيئة تطويرية Integrated Development Environment (IDE) تحوي كل الأدوات.
- إنشاء برامج مساعدة لا تعتمد على واجهات (Interface) تتميز بأنها سرية وآمنة وقوية.
- مكتبات تنفيذ المشروع.
- MFC صفوف ميكروسوفت التأسيسية Microsoft foundation class تساعد في إنشاء الواجهات Graphic user interface (GUI) (واجهات التخاطب مع المستخدم).
- أدوات البناء Built tools تساعد في استخدام الأدوات الموجودة في C++.

الشكل العام للبرنامج:

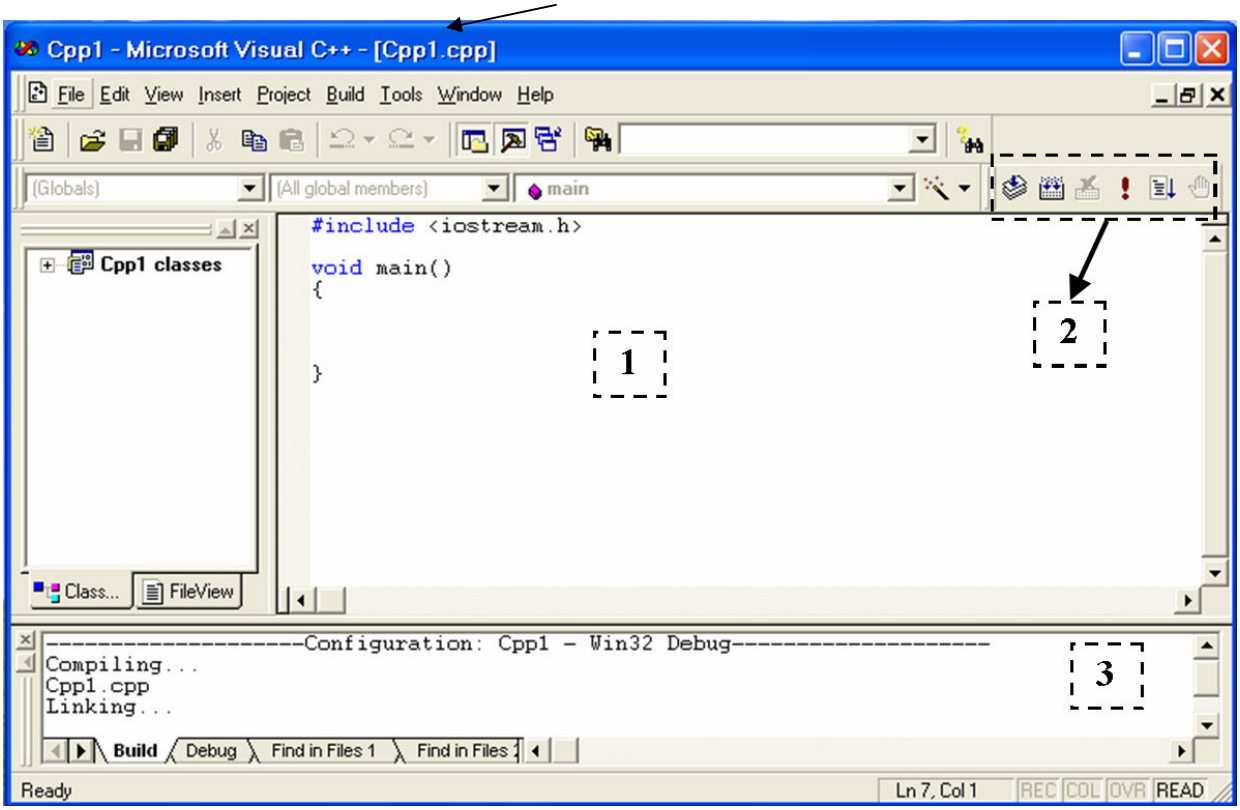
Header Files	1.	#include <library_name.h>	١. استيراد المكتبات
	2.	Public Declaration	٢. منطقة التصاريح العامة
Program Body	3.	Main ()	٣. الدالة الرئيسية
	4.	{	٤. بداية الدالة الرئيسية
	5.	Private Declaration	٥. منطقة التصاريح الخاصة
	6.	Statements.. Statements.. Statements..	٦. جمل برمجية
	7.	}	٧. نهاية الدالة الرئيسية

الفصل
3

واجهة بيئة C++
C++ Interface

واجهة البرنامج:

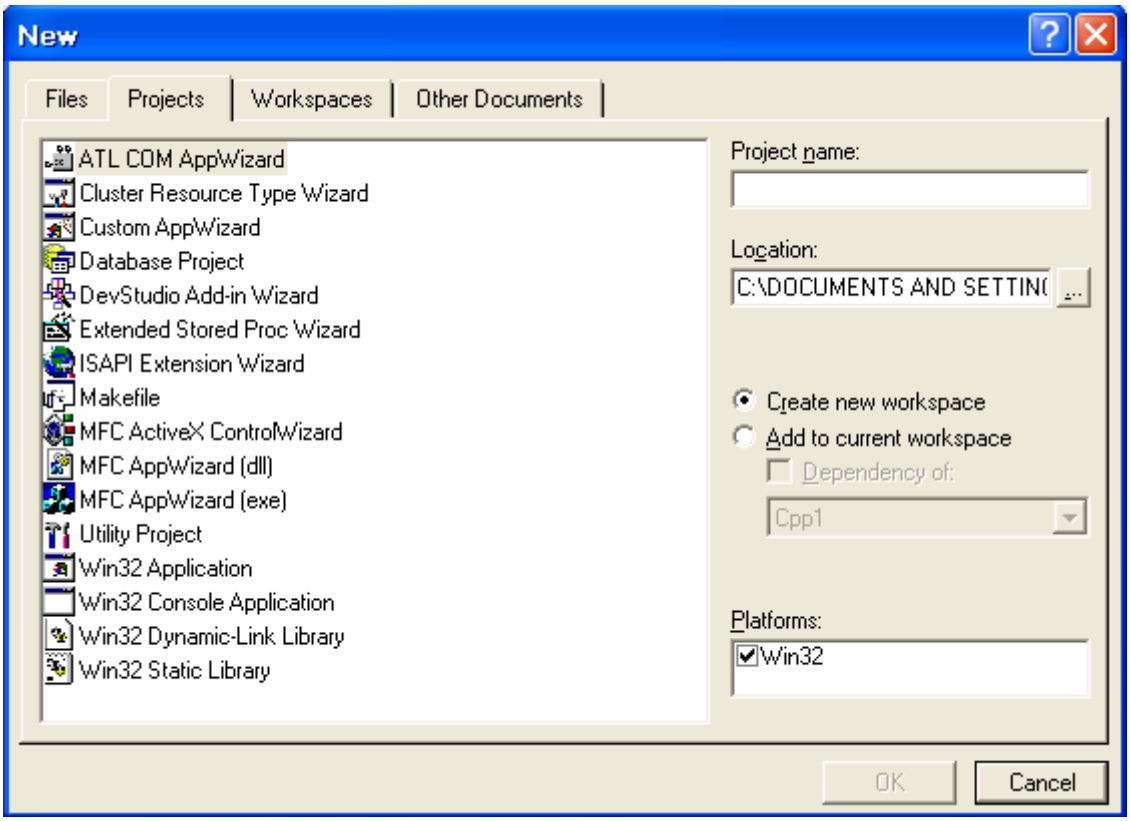
اسم الملف المصدري مع الامتداد cpp.



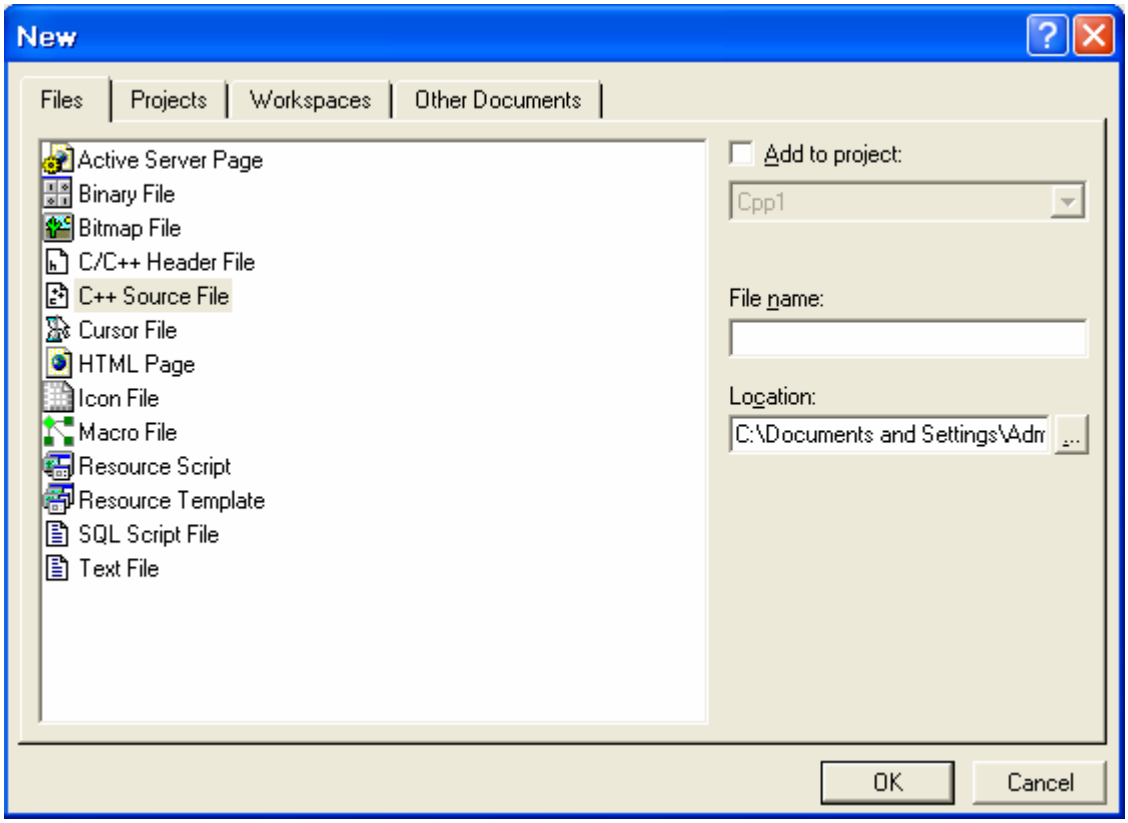
أهم الأجزاء:

[١] مكان كتابة الكود.
[٢] تنفيذ وعرض البرنامج، ويتكون من مرحلتين: مرحلة تكوين ملف مصدري، ثم ترجمة البرنامج للتنفيذ.
[٣] منطقة عرض الأخطاء، ومن خلالها يتم تحديد مكان الخطأ Error مع نوعية الخطأ.

أنواع المشاريع:



PROJECTS	المشروعات
Database Project	قواعد بيانات
MFC ActiveX ControlWizard (ocx)	ملفات مساعدة (لها خصائص وليس لها واجهات)
MFC AppWizard (dll)	ملفات مساعدة (لها خصائص مثل النموذج) تساعد في تصميم الواجهات.
MFC AppWizard (exe)	ملفات مساعدة (تحوي النوعين dll, ocx)
Utility Project	برامج خدمية
Win32 Application	تصميم تكوين مشروع من عدة تطبيقات
Win32 Console Application	تطبيق شاشة سوداء



FILES	الملفات
Active Server Page	صفحات انترنت تفاعلية ASP
Binary File	ملفات ثنائية (0/1)
Bitmap File	خريطة بيتات تنتج صورة
C/C++ Header File	مكتبات "إنشاء مكتبات" بامتداد h.
C++ Source File	برامج C++ "التي سنستخدمها" ملفات مصدرية
Cursor File	إنشاء صور مؤشر الفأرة
HTML Page	إنشاء صفحات ويب في بيئة html
Icon File	إنشاء أيقونة (رمز)

الفصل
4

مكونات C++
C++ Components

المكتبات:

١) iostream.h (مكتبة عامة لأوامر الإدخال والإخراج)
٢) stdio.h (مكتبة عامة "أقدم مكتبة" لأوامر الإدخال والإخراج)
٣) conio.h (مكتبة دوال أوامر الشاشة)
٤) math.h (مكتبة الدوال الرياضية)
٥) String (مكتبة دوال معالجة النصوص)

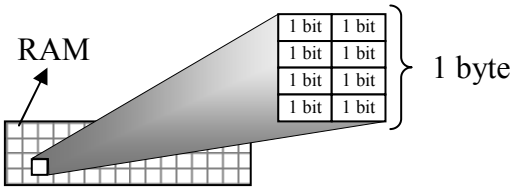
ملاحظة:

في أسماء المكتبات مثل (iostream.h , stdio.h) :
- i : يعني أوامر الإدخال ودوال الإخراج (Input).
- o : يعني أوامر الإخراج ودوال الإدخال (Output).
- s : البعض يقصد بها Standard والبعض يصفها بـ System.

أنواع البيانات:

INTEGER	Bytes	REAL	Bytes	STRING	Bytes	LOGIC	Bytes
Short	2	Float	4	Char	1	Bool	1
Int	4	Double	8	String	8		
Long	4						

هذه المسميات موجودة في الذاكرة العشوائية (RAM)، لكل نوع تقسيم "حجم" معين.



النوع	الحجم
Bit	2 (0/1)
Byte	8 bit
Kilobyte	1,000 byte
Megabyte	1,000,000 byte
Gigabyte	1,000,000,000 byte

المتغيرات:

أسماء كلمات ليست من كلمات اللغة المحجوزة، يصطلحها المبرمج (مستخدم بيئة تطوير لغة البرمجة) بغرض تخزين قيم في الذاكرة الرام لتمكين المبرمج من إجراء العمليات المختلفة على المتغيرات. فالمتغير يحفظ في موقع في الذاكرة، وإذا أراد المبرمج أن يزيد أو ينقص قيمة المتغير فيستطيع من خلال اسم المتغير.

الشكل العام:

```
DataType VariableName;
```

أمثلة

```
1. int x;
2. char ch;
```

إسناد قيم للمتغيرات:

1. int x = 5 ;	تعريف متغير x يحمل قيمة ابتدائية 5.
2. x = 10 ;	إسناد قيمة جديدة لـ x.
3. x = 20 ;	تغيير القيمة السابقة بقيمة جديدة أخرى.
4. x = 3 + 5 ;	إسناد ناتج القيمة الحسابية[1] للمتغير (المتغير سيحمل القيمة ٨)
5. cin >> x ;	تغيير القيمة أثناء التشغيل من قبل المستخدم[2].
6. char ch = ' y ' ;	إسناد قيمة حرفية لمتغير (يكتب داخل تعليق مفرد ' ')

شروط تعريف المتغيرات:

١. لا يبدأ برقم أو عملية حسابية أو رمز ما عدا _ (underscore).
٢. ألا يحتوي على عملية حسابية أو رمز أو فراغ.
٣. ألا يزيد عن 255 حرفاً.

(1) العمليات الرياضية في الفصل السابع.
(2) استقبال القيم من المستخدم أثناء التشغيل في الفصل التالي.

القيم الابتدائية الثابتة والمتغيرة (Initialization & Const)

القيمة الابتدائية :

هي قيم تسند للمتغير بمجرد تعريفه وهي نوعين:

القيم المتغيرة :

يمكن تغييرها في البرنامج من قبل المبرمج أو أثناء التشغيل "run mode" باستخدام "cin".

القيم الثابتة (const):

لا يمكن تغييرها بأي حال من الأحوال.

فائدتها : حماية القيم التي نحتاجها كما هي ولا نريد أن يتم تغيير قيمتها بالخطأ.
مثل : قيمة الثابت π (3.14)

مثال لمتغير:

1. int x = 5 ;	تعريف متغير x يحمل قيمة ابتدائية 5.
2. x = 10 ;	إسناد قيمة جديدة لـ x.
3. x = 20 ;	تغيير القيمة السابقة بقيمة جديدة أخرى.
4. cin >> x ;	تغيير القيمة أثناء التشغيل من قبل المستخدم.

مثال لثابت:

1. const int x = 5;	تعريف ثابت يحمل قيمة ثابتة 5.
2. x = 10 ; // Error	إسناد قيمة جديدة يولد خطأ في تنفيذ البرنامج.

التعليقات Comments:

عبارة عن توضيحات يكتبها المبرمج لا تدخل في تركيب البرنامج (لا ينفذها المترجم).

وتكون على شكلين :

- تعليق السطر الواحد :

```
1. // This is a comment
2. // And this is another comment
```

- تعليق الأسطر المتعددة:

```
1. /*
2. This is a comment
3. In tow lines
4. */
```

```
1. /* This is
2. a comment
3. in three lines */
```

الفصل
5

أنواع المكتبات
Libraries type

المكتبة IOSTREAM

تم دمج مكتبتين ضمن هذه المكتبة :
- Istream مكتبة الإدخال.
- Ostream مكتبة الإخراج.

أوامر الإدخال والإخراج:

تحتوي مكتبة iostream.h على دوال منها cin و cout ويتم استخدامهما كالتالي:

إدخال قيمة للمتغير x من قبل المستخدم:	1. cin >> x ;
إخراج قيمة المتغير إلى الشاشة:	2. cout << x ;
أرقام ورموز وحروف (انجليزية) باستخدام شرطة مزدوجة " "	3. cout << " نص " ;
حرف واحد باستخدام شرطة مفردة ' '	4. cout << ' c ' ;

مثال ١:

```
1. #include <iostream.h>
2. Main()
3. {
4.     int x ;
5.     cin >> x ;
6.     cout << " X value is: " << x ;
7. }
```

مثال ٢:

```
1. #include <iostream.h>
2. Main()
3. {
4.     int x , y ;
5.     cin >> x >> y ;
6.     cout << " first value is: " << x  << " second value is: " << y ;
7. }
```

المحارف الخاصة:

هي رموز محجوزة تعبر عن الحروف غير المطبوعة وتستخدم مع الدوال مثل (cout) و (printf) وتكون ضمن إشارتي تنصيص مزدوجة أو مفردة.

المحرف	المعنى	توضيح
\n	New line	سطر جديد
\t	8 Spaces (Tap)	٨ مسافات فارغة
\b	Backspace	الرجوع للخلف
\a	Sound "beep"	إصدار صوت من الجهاز

مثال:

```
1  cout << '\n';

2  cout << "Ahmed \t 20";

3  cout << "khaled\nSaleh";
```

الناتج: النزول إلى سطر جديد فارغ

الناتج :

```
Ahmed        20
```

الناتج:

```
khaled
Saleh
```

دوال تقوم بعمل المحارف الخاصة:

تستخدم مع الدالة (cout) .

الدالة	المعنى	توضيح
endl	New line	سطر جديد
ends	8 Spaces (Tap)	٨ مسافات فارغة

مثال:

```
1  cout << "Ahmed" << ends << "20";

2  cout << "khaled" << endl << "Saleh";
```

الناتج :

```
Ahmed        20
```

الناتج:

```
khaled
Saleh
```

المكتبة Stdio.h :

تحتوي على دالتين مهمتين :

- printf وهي دالة خاصة بعمليات الإخراج.
- scanf وهي دالة خاصة بعمليات الإدخال.

١) PrintF :

تتميز printf عن cout أنه يمكن كتابة النص والمتغير في نفس السطر بدون الحاجة لمعامل الاخراج (<<) ولكن بدلاً عن كتابة اسم المتغير يكتب التمثيل الديناميكي للمتغير حسب نوعه مسبوقاً بالرمز %:

النوع		التمثيل الديناميكي	ملاحظة
int	←	%d	أول حرف من من digital
char	←	%c	أول حرف من char
string	←	%s	أول حرف من string
float	←	%f	أول حرف من float

الصيغة العامة:

طباعة نص:	Printf(" النص ");
طباعة نص مع متغير:	printf("التمثيل الديناميكي النص ", var-name);
طباعة أكثر من متغير:	printf("تمثيل ديناميكي تمثيل ديناميكي", var-name1, var-name2);

أمثلة:

1. Printf(" welcome ");	
2. int a = 255 ;	
3. printf(" Area = %d " , a);	Area = 255
4. int x = 1; int z = 2;	
5. printf(" v1 = %d and v2 = %d ", x , z);	v1 = 1 and v2 = 2

٢) ScanF :

يمكن من خلال هذه الدالة استقبال المتغيرات من المستخدم وإسنادها للمتغيرات تماماً مثل (cin).

الصيغة العامة:

استقبال قيمة من المستخدم:

```
scanf (" التمثيل الديناميكي ", &var-name);
```

استقبال قيمتين من المستخدم:

```
scanf ("تمثيل ديناميكي تمثيل ديناميكي", &var-name1, &var-name2);
```

ملاحظة:
يجب كتابة الرمز & قبل أي متغير.

أمثلة:

```
1.  int x ; char y;
2.  scanf ("%d "  ,  &x );
3.  scanf ("%d%c ",  &x  , &y );
```

واجب :
ابحث عن الدوال : ()getch و ()puts.

المكتبة Math.h :

تحتوي على دوال رياضية كثيرة مثل:

توضيح	الرمز الرياضي	الدالة
الأعداد الحقيقية	$\lvert x \rvert$	abs(x)
جا س		sin (x)
جتا س		cos (x)
ضا س		tan (x)
جا$^{-1}$ س		sinh (x)
جتا$^{-1}$ س		cosh (x)
ضا$^{-1}$ س		tanh (x)
الاس	x^y	pow (x , y)
e	e^x	exp (x)
الجذر	$\sqrt{x}$	sqrt (x)
اللوغاريتم	Log x	log (x)
تقريب الكسور للأعلى		ceil (x)
حذف الكسور		floor (x)

مثال:

```
cout << ceil( 3.44 );
cout << ceil( -3.77);
cout << floor( 3.44 );
```

الناتج:

```
4
-3
3
```

المكتبة String :

توفر نوع من أنواع البيانات وهو (string) الذي يقبل تخزين مجموعة حروف ورموز وأرقام كنص في متغير واحد.

تختلف المكتبة عن سابقاتها، فمن أجل تعريف متغير x من نوع string يجب :

١. تضمين المكتبة string.
٢. إلغاء h. من اسم المكتبة "باستثناء المكتبات القديمة ".
٣. تحديث المكتبات..

مثال:

```
1.  #include <string>
2.  #include <iostream>
3.  #include <stdio.h>
4.
5.  using namespace std;

6.
7.  main ()
8.  {
9.      string s ;
10.     s = " Bassam ";
11.     cout << s;
12. }
```

تضمين المكتبة string بدون كتابة اللاحقة h.
تضمين مكتبة جديدة لذا لا نكتب اللاحقة h.
تضمين مكتبة قديمة لذا تبقى اللاحقة مثل stdio.h, math.h

تحديث المكتبات حيث std يحوي أوامر جديدة للمكتبات القابلة للتحديث "المكتبات الجديدة"

تعريف متغير نصي
إسناد قيمة نصية للمتغير
طباعة المتغير (عرض المتغير على الشاشة).

الفصل
6

تنسيق مخرجات البرنامج
Format Outputs

تنسيق الشاشة:

يمكن تحديث المكتبات بدون تضمين مكتبة string ، ومن مميزات تحديث المكتبات الحصول على دوال system التي تمكننا من تنسيق شاشة الإخراج وكذلك استخدام جميع أوامر نظام (DOS).

مثال:

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  main ()
5.  {
6.      system("color f0") ;
7.      cout << "new colors";
8.      system("pause");
9.  }
```

تضمين مكتبة مع حذف اللاحقة h.
تحديث المكتبات

تنسيق لون النص إلى اسود (0) والخلفية إلى أبيض (F)
طباعة نص (سيظهر باللون الأسود)
جعل البرنامج في حالة انتظار

ملاحظات:

١. يتم تمثيل الألوان برقم "سادس عشري" من صفر إلى f حيث يمثل جميع الألوان الأساسية.
٢. عند كتابة رقم واحد "color 9" فهذا سيغير لون النص فقط.
٣. عند كتابة رقمين "color f0" فإن الأول سيغير لون النص والثاني سيغير لون الخلفية.
٤. عند كتابة رقمين متشابهين "color 99" فلن يتغير أي لون، باعتبار أن لون الخط سيشبه لون الخلفية ولذا لن يظهر شيء فلذلك يتم تجاهل الألوان وإعادة الألوان الافتراضية.

تنسيق الألوان الافتراضي لمحرر بيئة Microsoft C++:

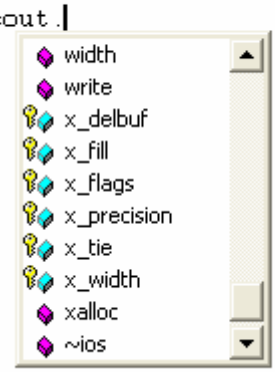
١. الكلمة المحجوزة تظهر بلون أزرق.
٢. الكلمة غير المحجوزة تظهر بلون أسود.
٣. التعليقات تظهر بلون أخضر.
٤. لغة C++ حساسة لحالة الأحرف (r لا يساوي R)

التنسيق باستخدام (cout):

تحتوي cout على العديد من الخواص الخاصة بالتنسيق ومنها :

- width والتي تعمل إزاحة لليمين من جهة اليسار بمقدار معين
- Fill والتي تقوم بتعبئة الفراغات التي تركتها width برمز معين.
- Precision والتي تقوم بتقريب الأرقام.

```
#include <iostream.h>
main(){
 cout.
}
```



مثال١:

```
1. cout.width(20);
2. cout << "Welcome";
```

النتيجة:

Welcome

مثال٢:

```
1. cout.width(20);
2. cout.fill('#');
3. cout << "Welcome";
```

النتيجة:

##########Welcome

مثال٣:

الكود:

```
1.  cout.precision(3);
2.  cout << 3.449;
3.  cout.precision(2);
4.  cout << 3.449;
5.  cout.precision(3);
6.  cout << 3.482;
7.  cout.precision(2);
8.  cout << 3.482;
9.  cout.precision(1);
10. cout << 3.482;
11. cout.precision();
12. cout << 3.482;
```

النتيجة:

3.45

3.4

3.48

3.5

3

3

الفصل
7

العمليات في C++
C++ Operations

العمليات الحسابية:

الرمز	التوضيح
+ - * /	العمليات الرياضية
y / x	القسمة
y \ x	القسمة الصحيحة
y % x	باقي القسمة

مثال :

```
1.  int x = 3 ;
2.  int z = 7 ;
3.
4.  cout <<  "z + x= " ;
5.  cout << z + x ;
```

الناتج هو "z + x= "
الناتج هو 10

ملاحظة : ما داخل الأقواس 'المفردة' أو "المزدوجة" يعتبر نص.

عمليات المقارنة:

الرمز	التوضيح
>	أكبر من
<	أصغر من
>=	أكبر أو يساوي
<=	أصغر أو يساوي
==	يساوي
!=	لا يساوي

مثال :

```
1.  int x = 3 ;
2.  int z = 7 ;
3.
4.  if ( x != z)
5.  {
6.     cout << "Not equal" ;
7.  }
```

إذا كان(1) x لا يساوي(2) z

فاطبع الجملة "Not equal"
ناتج البرنامج "Not equal" لأن ناتج الشرط True "صواب"

(1) جمل التحكم في الفصل الثامن.
(2) المساواة تعني "مقارنة قيمتين"، الأمثلة في الصفحة التالية.

العمليات المنطقية Logic Effects:

الرمز	التوضيح
&&	و
||	أو
!	نفي

مثال :

```
1.  int x = 3 ;
2.  int z = 7 ;
3.
4.  if ( x >0 &&  z>0)
5.  {
6.     cout << "Both numbers positive" ;
7.  }
```

إذا كان x أكبر من الصفر وأيضاً z أكبر من الصفر (ناتج الشرط True)

ناتج البرنامج " Both numbers positive"

المساواة والإسناد Equal and Assigned:

<u>الإسناد: هو إعطاء المتغير قيمة:</u>

مثال:

```
1   int x  ;
2   x = 7 ;

3   cout << x;
```

الناتج 7

<u>المساواة: هو مقارنة قيمتين:</u>

مثال:

```
1.  int x = 7  ;
2.  int y = 7 ;
3.
4.  cout << x==y;
5.  cout << x > 3
6.
7.  int z = x == 8
8.  cout << z ;
```

الناتج صواب (True) "سيطبع 1 في الشاشة"
الناتج خطأ (False) "سيطبع 0 في الشاشة"

اسناد ناتج المقارنة للمتغير z
الناتج False لأن x تساوي 7 وليس 8
وستتم طباعة 0 على الشاشة.

ملاحظة:

x = = y تعني (هل أن x يساوي y) وهي عملية مقارنة ناتجها إما صواب أو خطأ.

تحويل المعادلات الرياضية إلى معادلات برمجية:

	المعادلة الرياضية	المعادلة البرمجية
1)	$z = x^2 + x + 7$	Z = pow(x , 2) + x + 7 ;
2)	$z = \frac{x+1}{y+1}$	Z = (x+1) / (y+1) ;
3)	$z = \frac{(x^2 + x + 7)^2}{y + x + 1}$	Z = pow((pow(x , 2) + x + 7) , 2) / (y + x + 1)

توجد الدالة pow ضمن المكتبة math.h لذلك يجب تضمين المكتبة math.h في البرنامج، المزيد من الدوال الرياضية في الفصل الخامس .

س: كيف تكتب المعادلة التالية برمجياً؟

$$y = \begin{cases} x+1 : x < 0 \\ x^2 + x + 7 : x > 0 \end{cases}$$

ج:

إذا كان x أصغر من الصفر

```
if ( x < 0 )
{
    Y = x + 1 ;
}
else
{
    Y = pow( x , 2 )  + x + 1 ;
}
```

وإذا كان غير ذلك[1] (جميع الحالات التي لا توجد في الشرط السابق مثل x>0 أو x=0)

س: كيف تكتب المعادلة التالية برمجياً؟

$$R = x^{y^2}$$

ج:

```
R = pow( x, pow( y , 2 ) ) ;
```

س: كيف تكتب المعادلة التالية برمجياً؟

$$Y = \sqrt{3^2}$$

ج:

```
Y = sqrt( pow( 3 , 2 ) ) ;
```

أولوية العمليات الحسابية:

٦) ما بداخل الأقواس.
١- الأس.
٢- الضرب ثم القسمة.
٣- الجمع أو الطرح.

(1) جملة التحكم if لها عدة أشكال المزيد في الفصل الثامن.

برنامج لإيجاد مساحة المستطيل:

من المهم :
- فهم فكرة البرنامج.
- تحويل الفكرة إلى خطوات منطقية، وأهمها معرفة معادلة مساحة المستطيل.

مساحة المستطيل = الطول × العرض

إذاً نحتاج إلى ٣ متغيرات ، متغيرين سيقوم المستخدم بإدخالهما (الطول والعرض) ومتغير سيحتوي على ناتج ضرب المتغيرين (المساحة).

خطوات الحل البرمجي:

١) التصريحات Declare (تعريف المتغيرات).
٢) المدخلات Input (إسناد قيم للمتغيرات "الطول والعرض").
٣) المعالجة Process (الضرب).
٤) المخرجات Output (مساحة المستطيل).

```
1. int height, width, area;
2. cin >> height >> width;
3. area = height * width
4. cout << area;
5.
```

تعريف ثلاثة متغيرات من نفس النوع في سطر واحد :
استقبال قيمتين من المستخدم:
تخزين نتيجة ضرب القيمتين في متغير المساحة
عرض الناتج (مساحة المستطيل)
الناتج:

```
9
3
27Press any key to continue_
```

كما نلاحظ أن الناتج في المثال السابق غير واضح وعند تنفيذ البرنامج تظهر شاشة سوداء لا يوجد فيها أي معلومات تساعد مستخدم البرنامج في معرفة ما المطلوب منه وماذا يجب أن يعمل..

تنسيق المخرجات:

أعادة المثال السابق:

```
1. int height, width, area;
2. cout << "Enter Height: ";
3. cin >> height;
4. cout << "Enter Width: ";
5. cin >> width
6. area = height * width
7. cout << "---------\n";
8. cout << "Resutl is: " <<area << endl;
```

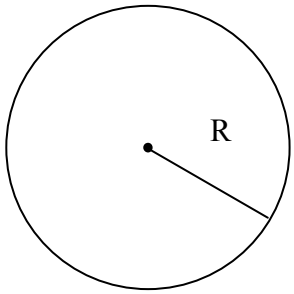
طباعة نص يطلب إدخال الطول
استقبال الطول من المستخدم
طباعة نص يطلب إدخال العرض
استقبال العرض من المستخدم

طباعة خط أفقي وسطر جديد
طباعة "النتيجة هي" ثم ناتج الضرب ثم سطر جديد
الناتج:

```
Enter Height: 7
Enter Width: 8
---------
Result is: 56
Press any key to continue_
```

الواجب:

أكتب برنامج لحساب مساحة الدائرة إذا علمت أن:



$$\text{Circle} = \pi * R^2$$

حيث أن :
R : نصف القطر (معطى).
π : 3.14 (ثابت).

الفرق بين signed و unsigned:

signed جعل نوع البيانات يقبل القيم السالبة وهو الافتراضي حتى إذا لم يكتب، بينما unsigned لا يقبل القيم السالبة ، حيث يرفض أي قيمة سالبة وتظل قيمته عشوائية إلى أن يتم إسناد قيمة موجبه إليه.

مثال:

```
1.    #include <iostream.h>
2.
3.    main ()
4.    {
5.        signed int x;
6.        unsigned int z;
7.        x = 100;
8.        z = 100;
9.        cout << x << endl;
10.       cout << y << endl;
11.       x = -100;
12.       z = -100;
13.       cout << x << endl;
14.       cout << y << endl;
15.   }
```

الناتج

```
100
100

-100
4294967196
```

والرقم 4294967196 عبارة عن رقم عشوائي جاء من الذاكرة نتيجة لأن المتغير y لم يسند له أي قيمة، وذلك لأن النوع unsigned يجعل المتغير يرفض أن يسند إليه قيمة سالبة.

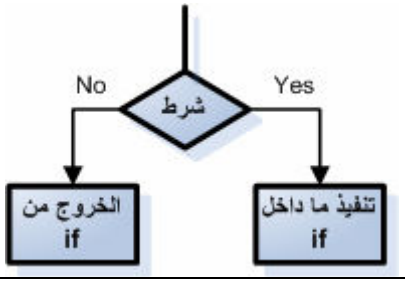
الفصل
8

جمل التحكم
Control Statements

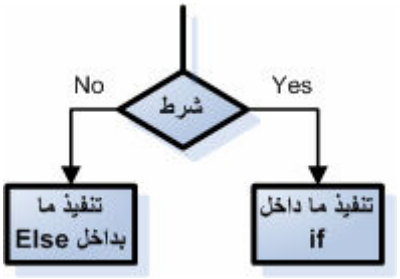
الجمل الشرطية Condition Statements:

١) جملة IF:

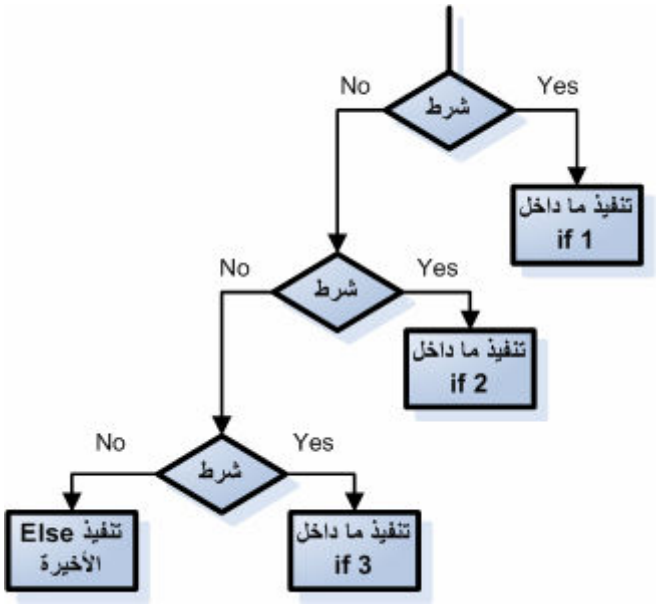
```
if ( Expression )
{
    Statements;
    Statements;
    .....;
}
```




```
if ( Expression )
{
    Statements;
    Statements;
    .....;
}
else
{
    Statements;
    Statements;
    .....;

}
```

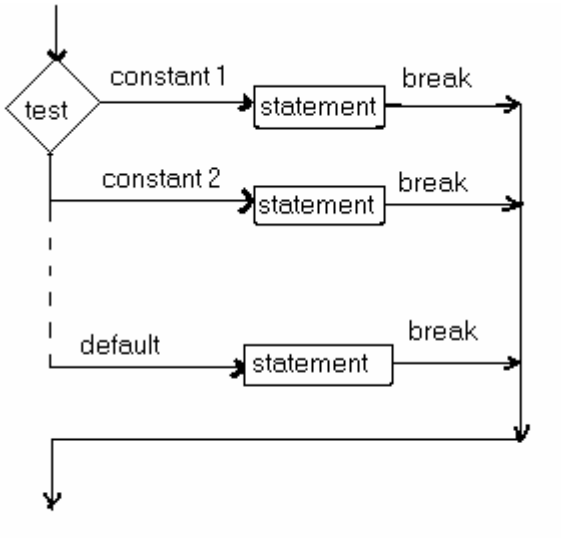



```
if ( Expression ) {
    Statements;
    Statements;
    .....;
}
else if ( Expression ) {
    Statements;
    Statements;
    .....;

}
else {
    Statements;
    Statements;
    .....;

}
```



٢) جملة switch:

```
switch ( var )
{
    case 1 :
        Statements1;
    break;

    case 2:
        Statements;
    break;

    default:
        Statements;
}
```



ملاحظة :
فائدة break بعد كل case أنها توقف عمل switch بعد تنفيذ جملة case المناسبة، وإذا لم يتم كتابة الكلمة break فإن البرنامج سينتقل إلى case التالية وينفذها حتى لو لم ينطبق الشرط var عليها.
يمكن كتابة الجزء الأخير الخاص بـ default ويمكن عدم كتابته، حيث يتم تنفيذ الجمل داخل default عندما لا تتحقق أي من الحالات "cases" السابقة، فهي تشبه else الأخيرة في جملة if.

مثال:

```
1.  int r = 3
2.  switch ( var )
3.  {
4.      case 1 : cout << "one\n";     break;
5.      case 2:  cout << "two\n";     break;
6.      case 3:  cout << "three\n";     break;
7.      default: cout << "Error!\n";
8.  }
```

ملاحظة:
إذا كان المتغير حرفي نستخدم علامة الاقتباس المفردة مثل: case 'y' وإذا كان نصي نستخدم علامة الاقتباس المزدوجة مثل: case "yes".

الواجب :

اكتب برنامج لمعرفة نوعية العدد (سالب أو موجب أو غير ذلك).

دوال الدوران Loops Functions:

١) For:

تحتاج دالة for إلى عداد (رقم تبدأ منه الدوران ورقم تنتهي إليه) لكي تنفذ الدورات ومقدار الزيادة[1].

الشكل العام Public formula:

```
for ( initialization_value; condition; Increment or Decrement)
    Statements...
```

مثال :

الكود:

```
1.  for ( int i = 0; i <=3; i++)
2.  {
3.      cout << "i value is : " << i << endl;
4.  }
```

النتيجة:

```
i value is: 0
i value is: 1
i value is: 2
i value is: 3
```

٢) While:

تحتاج دالة while إلى شرط يحدد استمرارها أو توقفها، فهي ستستمر بلا توقف طالما الشرط متحقق.

الشكل العام Public formula:

```
while (condition)
    Statements...
```

مثال :

الكود:

```
1.  int w = 3;
2.  while ( w <=3 )
3.  {
4.      cout << "value is : " << w << endl;
5.      w++;
6.  }
```

النتيجة:

```
value is: 0
value is: 1
value is: 2
value is: 3
```

ملاحظة: يمكن الاستغناء عن الأقواس {} الخاصة بدالة for و while و if إذا كانت الجملة التي تنفذها تتكون من سطر واحد.

(1) مقدار الزيادة أو النقصان في الصفات القادمة.

٣) do while:

الشكل العام Public formula:

```
do
{
    Statements...
}
while (condition)
```

مثال :

الكود:

```
1.  int w = 0;
2.  do
3.  {
4.      cout << "value is : " << w << endl;
5.      w++;
6.  }
7.  while ( w <=3 );
```

النتيجة:

```
value is: 0
value is: 1
value is: 2
value is: 3
```

لكن do while تقوم بتنفيذ الكود مرة واحدة حتى لو كان الشرط خاطئاً:

الكود:

```
1.  int w = 0;
2.  do
3.  {
4.      cout << "value is : " << w << endl;
5.  }
6.  while ( w >0 );
```

النتيجة:

```
value is: 0
```

القيمة التزايدية Increment value (معنى ++i):

تعني زيادة المتغير i بمقدار واحد فقط وهو اختصار للجملة التالية:

```
i = i + 1;
```

ويمكن زيادة المتغير i بأي مقدار نريد بالشكل التالي:

```
i += 5;
```

وهذا يكافئ السطر التالي:

```
i = i + 5;
```

أو إنقاص المتغير i بأي مقدار نريد بالشكل التالي:

```
i -= 5;
```

وهذا يكافئ السطر التالي:

```
i = i - 5;
```

الفرق بين ++i و i++ :

مثال ١:

طباعة i ثم زيادته بمقدار واحد.	`cout << i ++;`
زيادة i بمقدار واحد ثم طباعته.	`cout << ++ i;`

مثال ٢:

`int z , i ;`
`i = 1 ;`
`z = 5 * i + ++ i ;`
	`int z , i ;`
	`i = 1 ;`
	`z = 5 * i + i ++ ;`

التنفيذ : `z = 5 * 1 + 2;`	التنفيذ : `z = 5 * 1 + 1;`
قيمة z: 7	قيمة z: 6

الواجب:

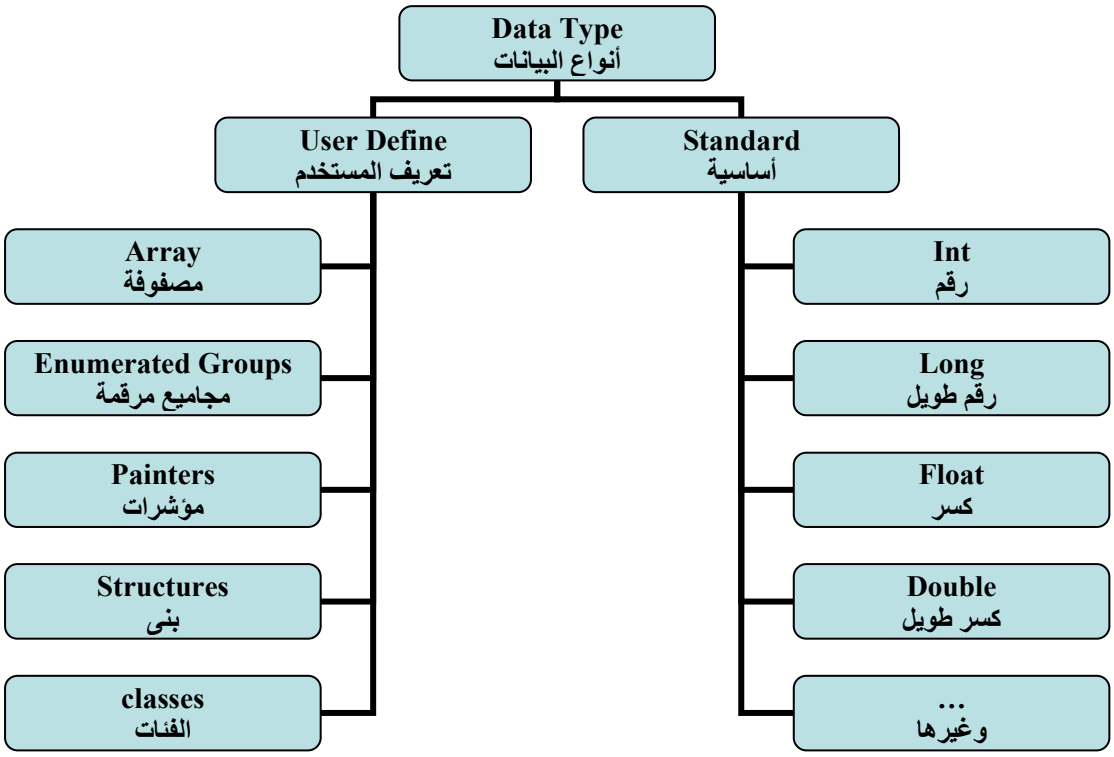
عمل برنامج يقم بطباعة مثلثات باسكال باستخدام دالتي for فقط.

١
٢ ٢
٣ ٣ ٣
٤ ٤ ٤ ٤
٥ ٥ ٥ ٥ ٥

للتسهيل يمكن عملها باستخدام رمز النجمة (*) فقط:

*
* *
* * *
* * * *
* * * * *

أنواع البيانات
Data Types



الأنواع القياسية Standard:
هي أنواع بيانات معرفة مسبقاً يمكن استخدامها ولا يمكن تغييرها فلا يمكن زيادة حجمها أو تحديدها، أي لا يمكن للمستخدم التحكم بها.

الأنواع من تعريف المستخدم User Define:
هي أنواع بيانات يعرفها المستخدم بالطريقة التي يريد ويستطيع تغييرها والتحكم بمساحتها.

المجاميع المرقمة Enumerated groups:

int يعتبر من المجاميع المرقمة المعرفة مسبقاً، ويمكننا تعريف مجاميع خاصة بنا حسب الحاجة.

أمثله[1]:

int نوع بيانات معرف مسبقاً ولا يمكن تغييره.
months معرف من المستخدم ويمكن تغييره أو حذفه.

```
enum int {-32,700, ..... , 32.700}
enum months { jan, feb, ... , des}
```

أمثله:

تعريف x من نوع weekday
x لا يقبل إلا قيمة من نوع weekday
الناتج: 0

الناتج: 1

```
1. enum weekday { sat, sun, ...., friday}
2. weekday x;
3. x = sat ;
4. cout << x;
5. x = sun ;
6. cout << x;
```

إعادة تعريف البيانات Data definition return :

أنواع البيانات مثل int, float, double يمكن إعطائها مسميات أخرى للتبسيط أو للحماية حيث إذا تم تغيير نوع البيانات فلن يعرف من يطلع على الكود ما هو هذا النوع.
يمكن تغيير اسم نوع البيانات من خلال الدالة TypeOf مع ملاحظة أن هذا التغيير لا يؤثر على نوع البيانات الأصلي.

إطلاق اسم جديد على النوع float (الأعداد الكسرية).
إسناد قيمة للنوع الجديد (أصبح fl يقوم بعمل float)
الناتج : 5.4

```
1. TypeOf float fl ;
2. fl x = 5.4 ;
3. cout << x;
```

(1) تم التعويض بالنقاط (sun, …., friday) لاختصار المثال.

المصفوفات Array:

جاءت المصفوفات لحل مشكلة الحاجة لإدخال عدد كبير من البيانات، فبدلاً من استخدام عدد كبير من المتغيرات لحفظ البيانات يتم استخدام المصفوفة التي تستطيع الاحتفاظ بالبيانات كمتغير واحد.

تعريفها:
مجموعة من المواقع المتجاورة في الذاكرة ولها نفس نوع البيانات وتستخدم لخزن البيانات.

	x
3	int
2	int
1	int
0	int
index	

أنواع المصفوفات Array types:

١. مصفوفات أحادية البعد Single Dimensional.
٢. مصفوفات متعددة الأبعاد Multi Dimensional.

مميزات المصفوفات:

١) تقليل حجم البرنامج.
٢) سهولة اسناد القيم واسترجاعها.
٣) استخدام تقنيات البحث والترتيب.
٤) الوصول المباشر Direct Access إلى البيان، وهذه ميزة غير موجودة في أي نوع آخر.

عيوب المصفوفات:

١) يمكن للمستخدم تحديد حجم المصفوفة عند تعريفها فقط (لا يمكن أثناء التشغيل تحديد حجمها).
٢) يجب أن تحتوي جميع القيم على نوع واحد من البيانات (لا يمكن تخزين بيانات من أنواع مختلفة).

الصيغة العامة Public formula:

١. مصفوفة أحادية:

```
Data_Type Array_name [ Array_Size ];
```

مثال:

```
int x[5];
```

حجم المصفوفة يتحدد بنوعها فإذا كانت رقمية int فإن كل عنصر فيها سيحجز 4 بايت، وبالتالي فإن حجم المصفوفة الأحادية سيكون بضرب حجم نوع البيانات في عدد صفوف المصفوفة كالتالي:

x
4 byte
4 byte
4 byte
4 byte
4 byte

4 byte * 5 rows = 20 bytes

وهذه هي المساحة المحجوزة للمغير x في الذاكرة

٢. مصفوفة ثنائية:

```
Data_Type Array_name [ Array_Rows_Size ][ Array_Cols_Size ];
```

مثال:

```
int x[3][4];
```

x

4 byte	4 byte	4 byte	4 byte	4 byte
4 byte	4 byte	4 byte	4 byte	4 byte
4 byte	4 byte	4 byte	4 byte	4 byte

حجم المصفوفة متعددة الأبعاد يكون بضرب حجم نوع البيانات × عدد صفوف المصفوفة × عدد أعمدة المصفوفة:

4 byte * (3 rows * 4 cols) = 48 bytes

إدخال البيانات للمصفوفة (الإسناد) [1]**:**

الطريقة الأولى:

تعريف مصفوفة وإسناد كل بياناتها في نفس الوقت:

```
int x[5] = { 1, 7, 10, 2, 5};
```

بقية الخانات ستكون صفرية:

```
int y[10] = {3, 5};
```

كل الخانات ستكون أصفار:

```
int z[4] = {0};
```

الطريقة الثانية:

تعريف مصفوفة:

```
int a[3];
```

إسناد قيمة للخانة الأولى في المصفوفة:

```
a[0] = 1;
```

إسناد قيمة للخانة الثانية في المصفوفة:

```
a[1] = 7;
```

إسناد قيمة للخانة الثالثة في المصفوفة:

```
a[2] = 10;
```

10
7
1

ويمكن إدخال البيانات إلى المصفوفة أثناء تشغيل البرنامج عن طريق (cin) :

إدخال جميع قيم المصفوفة باستخدام دالة For

```
int a[2];
for ( int i=0; i<=2; i++)
{
    cin >> a[i];
}
```

وبالمثل عملية الإخراج:

طباعة جميع قيم المصفوفة باستخدام دالة For

```
for ( int i=0; i<=2; i++)
{
    cout << a[i] << endl;
}
```

(1) عند تحديد حجم المصفوفة بخمسة في لغة C++ فإن عدد العناصر خمسة أما في لغة Visual Basic فإن عدد العناصر سيكون ستة.

إدخال بيانات من النوع الحرفي char:

النوع char يقبل حرف واحد فقط ويحجز بايتاً واحداً في الذاكرة

تعريف وإسناد معاً

```
char c;
char c = 'a';
```

كلا الطريقتين الآتيتين تؤدي نفس الغرض في الإدخال إلى المصفوفة:

- إسناد كل حرف على حدة

```
char x[4] = { 'S', 'A', 'B', 'A'};
```

- إسناد كل الحروف دفعة واحدة.

```
char x[4] = "SABA";
```

A
B
A
S

ومن هنا نستنتج أن النوع String ما هو إلا مصفوفة حرفية وحجمها 255 بايتاً.

النوع string يستخدم نفس طريقة الإسناد.

```
string = "SABA";
```

ملاحظة :
عند تعريف وإسناد المصفوفة في نفس الوقت فإن لغة C++ تحدد حجم المصفوفة تلقائياً بمعرفة العناصر داخل الأقواس. أما إذا كنت ستعرف المصفوفة وتسند القيم فيما بعد فتأكد أن القيمة داخل قوسي المصفوفة [ ] تساوي عدد عناصر المصفوفة.

مثال:

```
int x[] = {1,2,3,4};
```

استخدام المصفوفة:

جمع قيمتين في المصفوفة وإسناد الإجمالي إلى متغير آخر:

```
1. int z;
2. int x[5] = { 1, 7, 10, 2, 5};
3. z = x[0] + x[3];
```

جمع جميع قيم عناصر المصفوفة:

```
1. int sum = 0;
2. for ( int i =0; i<5; i++)
3. {
4.     sum = sum + x[i];
5. }
6. cout << sum;
```

ضرب جميع قيم عناصر المصفوفة:

```
1. int m = 1;
2. for ( int i =0; i<5; i++)
3. {
4.     m = m * x[i];
5. }
6. cout << m;
```

المصفوفة متعددة الأبعاد:

- ثنائية البعد:

```
Data_Type Array_name [ x ][ y ];
```

- ثلاثية البعد:

```
Data_Type Array_name [ x ][ y ][ z ];
```

- رباعية البعد:

```
Data_Type Array_name name [ x ][ y ][ z ][ t ];
```

وهكذا يمكن إضافة أبعاد بحسب الحاجة.

مثال:

```
char ary[3][3] = {{'A','B','C'},{'D','E','F'},{'G','H','I'}};
```

A	B	C
D	E	F
G	H	I

إدخال البيانات:

كلا الثلاثة الطرق الآتية تؤدي نفس الغرض في الإدخال إلى المصفوفة الثنائية:

- الطريقة الأولى:

```
1.  int x[2][2] = {1, 5, 4, 9};
```

- الطريقة الثانية:

```
1.  int x[2][2] = { {1, 5} , {4, 9} };
```

- الطريقة الثالثة:

```
1.  int x[0][0] = 1;
2.  int x[0][1] = 5;
3.  int x[1][0] = 4;
4.  int x[1][1] = 9;
```

يمكن إدخال بيانات المصفوفة الثنائية باستخدام دالتي For:

```
1.  int x[2][2];
2.
3.  for ( int i =0; i<2; i++)
4.      for ( int r =0; r<2; r++)
5.          cin >> x[i][r];
```

وكذلك الإخراج:

```
1.  for ( int i =0; i<2; i++)
2.      for ( int r =0; r<2; r++)
3.          cout << x[i][r];
```

العمليات على المصفوفات:

- الإضافة.
- الحذف.
- البحث.
- الترتيب.
- التعديل
- وغيرها...

أ) البحث:

١- البحث عن قيمة موجودة في المصفوفة:

يتم البحث عن قيمة موجودة في المصفوفة بمقارنة القيمة المراد البحث عنها في عناصر المصفوفة.

مثال:

```
1.  int x = 4;
2.  int a[] = {1,2,3,4,5,6};
3.  int t;
4.  for ( int i =0; i<6; i++) {
5.      if ( x == a[i]) {
6.          t = 1;
7.          break;

8.      } else {
9.          t = 0;
10.     }
11. }

12. if ( t = 1)
13.     cout << "number found\n";
14. Else
15.     cout << "number not found\n";
```

المرور على كل عناصر المصفوفة بدالة for
إذا تساوت قيمة x مع قيمة في المصفوفة
فيتم اسناد قيمة 1 للمتغير t
وكذلك الخروج من دالة for الى السطر رقم 12
باستخدام الكلمة Break.
وإذا لم تتساوى قيمة x مع قيمة في المصفوفة
فيتم إسناد قيمة 0 للمتغير t

* ستدور الدالة for أربع مرات حتى تجد العنصر الذي
قيمته 4، وعندها فإن السطر 6 سينفذ وسيكون t=1
وستقفز إلى السطر رقم 12 بواسطة الأمر Break
وعندها سيتم طباعة النص في السطر 13
وإذا لم تجده "نفرض أننا نبحث عن القيمة 8"
فسيتم طباعة النص في السطر 15

٢- البحث عن أكبر قيمة موجودة في المصفوفة:
يتم البحث بأخذ عنصر من المصفوفة ومقارنته ببقية العناصر وخزن القيمة الكبيرة في temp .

```
1.   int a[11] = {5,3,7,10,8,6,2,4,11,2,0};
2.   int temp=0;
3.   for (int j=0; j<11; j++){
4.       if ( a[j] > temp ){
5.           temp = a[j];
6.       }
7.   }
8.   cout << temp << endl;
```

ب) الترتيب:

```
1.  int a[] = {5,3,7,10,8,6,2,4,11,2,0};
2.  int temp = 0;
3.  for (int i =0; i< 11; i++) {
4.      cout << temp << "\t [" << ends ;
5.      for (int r=0; r< 11; r++) {
6.          if(a[r+1]<a[r]){
7.              temp = a[r];
8.              a[r] = a[r+1];
9.              a[r+1] = temp;
10.         }
11.         cout << a[r] << ends;
12.     }
13.     cout << "]" <<endl;
14. }
15. cout << "===============\n";
16. for (j=0; j<11; j++){
17.     cout << a[j] << ends;
18. }
19. cout << endl;
```

طباعة قيمة temp لغرض مشاهدة التغييرات

إذا كان العنصر التالي أكبر من السابق يدل المواقع السطور 6,7,8 عبارة عن تبديل مواقع القيم في خلايا المصفوفة .
وإذا كان العنصر التالي أصغر من السابق فلن يتم تبديل موقعه حتى لو لم يكن أصغر قيمة في المصفوفة فسينتظر لحين الدورة الثانية لـ For الأولى التي ستقوم بإعادة الترتيب وتحسين النتائج أكثر وأكثر، انظر موقع القيمة 0 في الصورة أدناه.

طباعة المتغير لغرض مشاهدة التغييرات

طباعة المصفوفة بعد الترتيب وهو ما يهمنا

```
0       [ 3 5 7 8 6 2 4 10 2 0 11 ]
11      [ 3 5 7 6 2 4 8 2 0 10 11 ]
10      [ 3 5 6 2 4 7 2 0 8 10 11 ]
8       [ 3 5 2 4 6 2 0 7 8 10 11 ]
7       [ 3 2 4 5 2 0 6 7 8 10 11 ]
6       [ 2 3 4 2 0 5 6 7 8 10 11 ]
5       [ 2 3 2 0 4 5 6 7 8 10 11 ]
4       [ 2 2 0 3 4 5 6 7 8 10 11 ]
3       [ 2 0 2 3 4 5 6 7 8 10 11 ]
2       [ 0 2 2 3 4 5 6 7 8 10 11 ]
2       [ 0 2 2 3 4 5 6 7 8 10 11 ]
====================
0 2 2 3 4 5 6 7 8 10 11
Press any key to continue_
```

حل آخر:

```
int a[] = {5,3,7,10,8,6,2,4,11,2,0};
int temp = 0;
for (int i =0; i< 11; i++) {
    cout << temp << "\t [" << ends ;

    for (int r=i+1; r< 11; r++) {
        if(a[i]>a[r]){
            temp = a[r];
            a[r] = a[i];
            a[i] = temp;
        }
        cout << a[r] << ends;
    }
    cout << "]" <<endl;
}
cout << "===============\n";

for (int j=0; j<11; j++){
    cout << a[j] << ends;
}
cout << endl;
```

```
0         [ 5 7 10 8 6 3 4 11 2 2 ]
0         [ 7 10 8 6 5 4 11 3 2 ]
2         [ 10 8 7 6 5 11 4 3 ]
2         [ 10 8 7 6 11 5 4 ]
3         [ 10 8 7 11 6 5 ]
4         [ 10 8 11 7 6 ]
5         [ 10 11 8 7 ]
6         [ 11 10 8 ]
7         [ 11 10 ]
8         [ 11 ]
===============
0 2 2 3 4 5 6 7 8 10 11
Press any key to continue
```

ج) التعديل:

س: إذا علمت أن المصفوفة التالية تمثل مجموع درجات الطلاب، فالمطلوب إضافة 5 درجات للطلاب الذين تنقصهم 5 درجات للنجاح.

int Stud[6] = {70, 50, 45, 43, 47, 90};

فالمطلوب عمل الكود اللازم.

ج:

```
1.  for ( int i =0; i<6; i++) {
2.      if (Stud [i] < 50 && Stud[i] > 45) {
3.          // Stud[i] += 5;   // A
3.          // Stud[i] = 50;   // B
4.      }
5.  }
6.  for (  i =0; i<6; i++)
7.      cout << Stud[i] << ", " << ends;
8.  cout << endl;
```

طريقة A: زيادة الطالب 5 درجات.

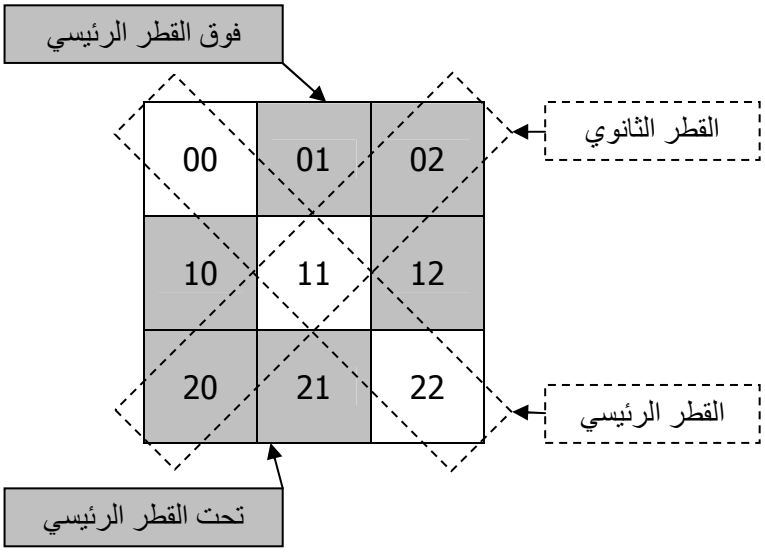
طريقة B: إعطاء الطالب درجة النجاح فقط.

الغي التعليق من أحد الطريقتين A أو B في في السطور رقم 3 لعرض الناتج بالطريقة التي تختارها.

الناتج (بالطريقة الثانية B):

70, 50, 45, 43, 50, 90

الواجب:



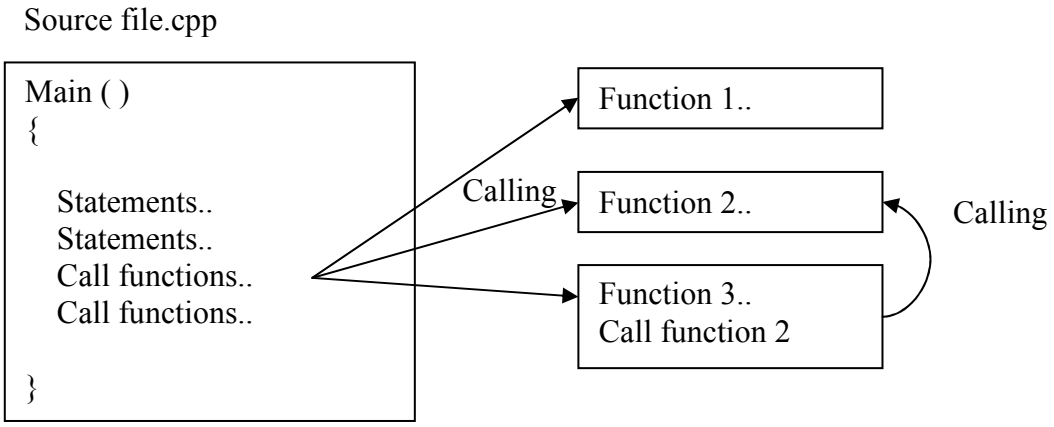
في الشكل أعلاه: المطلوب عمل برنامج يقوم بالتالي:

- طباعة عناصر القطر الرئيسي. (مساعدة: يتساوى فيه رقم الصف مع رقم العمود)
- طباعة عناصر القطر الثانوي. (مساعدة: رقم الصف + رقم العمود = حجم المصفوفة - ١)
- طباعة عناصر فوق القطر الرئيسي. (مساعدة: رقم الصف أكبر من رقم العمود)
- طباعة عناصر تحت القطر الرئيسي. (مساعدة: رقم الصف أصغر من رقم العمود)

الدوال Functions:

وهي عبارة عن برامج فرعية تشبه البرنامج الرئيسي main ، البرنامج main هو عبارة عن دالة تتميز بأن بيئة C++ تقوم بتشغيلها ، وتتولى دالة main تشغيل ما بداخلها واستدعاء الدوال الفرعية وتشغيلها.

تعريفها:
- مجموعة من التعليمات والأنشطة المكتوبة داخل برنامج مستقل (فرعي) يتم استدعائها داخل البرنامج الرئيسي.



ملاحظة:
استدعاء الدوال وتفعيلها يجب أن يكون من داخل الدالة الرئيسية main .

مميزات الدوال:
١- تقسيم البرامج الكبيرة إلى أجزاء صغيرة داخل البرنامج "Source file".
٢- اختصار الكثير من السطور في اسم الدالة (ذات الطابع التكراري).
٣- تنظيم البرنامج.
٤- قابلية استخدامها أكثر من مرة.
٥- سهولة التطوير والتعديل واكتشاف الأخطاء.

أنواع الدوال:

- دوال قياسية Standard function:
كل الدوال الموجودة في المكتبات مثل ()pow و cout ولا يمكن تعديلها.

- دوال من تعريف المستخدم User define function:
الدوال التي يقوم المستخدم بتكوينها.

أشكال الدوال:

- **إجراء Procedures:**
وهي دالة تقوم بعمل معين وتنفذه أو تطبعه على الشاشة، و تسمى (إجراء) لأنها لا تعيد قيمة.

- **دالة Function:**
وهي دالة تقوم بعمل معين وتعيد قيمة ويمكنها أن تطبع شيء على الشاشة وتعيد قيمة في نفس الوقت، ويمكن إسناد هذه القيمة إلى متغير ثم طباعته على الشاشة.

الصيغة العامة للإجراء:

التصريح	void FunctionName(parameters)
البناء	{
	Statements...
	{

الصيغة العامة للدالة:

- استخدام الكلمة المحجوزة return لإعادة قيمة:

```
DataType FunctionName(parameters)
{
   Statements...
   return value;
{
```

- استخدام نفس اسم الدالة لإعادة قيمة:

```
DataType FunctionName(parameters)
{
   Statements...
   FunctionName = value;
{
```

يمكن أن يحتوي الإجراء أو الدالة على باراميترات Parameters ويمكن ألا تحتوي عليها، فهذا يرجع للمبرمج.

مثال لإجراء:

```
1.  void sum(int a, int b)
2.  {
3.      cout << a + b;
4.  {
```

استدعاء الإجراء داخل البرنامج:

الناتج: 15	sum(7 , 8);

ملاحظات:

- لا يمكن إسناد الإجراء لمتغير فهو لا يعيد أي قيمة.
- لا يمكن إدخال الإجراء ضمن عمليات حسابية.

مثال لدالة:

```
1.  int sum(int a, int b)
2.  {
3.      Return (a + b);
4.  {
```

استدعاء وطباعة الدالة داخل البرنامج "سيتم طباعة ناتج الدالة":

```
cout << sum( 7 , 8 );
```

الناتج: 15

أو استدعاء وإسناد الدالة إلى متغير "سيتم إسناد ناتج الدالة لمتغير":

```
1.  int s;
2.  s = sum( 7 , 8 );
3.  cout << s;
```

الناتج: 15

استدعاء وطباعة الدالة داخل البرنامج وإدخالها ضمن عملية حسابية "سيتم في العلمية الحسابية التعامل مع ناتج الدالة كقيمة":

```
1.  cout << sum( 7 , 8 ) + 5;
```

الناتج: 20

استدعاء وإسناد الدالة إلى متغير وإدخالها ضمن عملية حسابية.. "سيتم إضافة ناتج الدالة إلى العملية الحسابية":

```
1.  s = sum( 7 , 8 ) + 5;
2.  cout << s;
```

الناتج: 20

تقديم نوع البيانات int قبل اسم الدالة (int function_name) يعني أن الدالة تعيد قيمة من النوع int بينما تقديم الكلمة void قبل اسم الدالة (void function_name) يعني أن الدالة لا تعيد أي قيمة من أي نوع.

تعريف الدوال :

- يمكن تعريف الدوال قبل الدالة الرئيسية main (التصريح والبناء معاً).

```
1.  int sum(int a, int b)
2.  {
3.      return a + b;
4.  {
5.
6.  void main()
7.  {
8.      cout << sum( 5 , 4 );
9.  }
```

- يمكن تعريف الدوال تحت الدالة الرئيسية main بشرط التصريح عنها قبل الدالة.

```
1.  int get_sum( int , int );
2.
3.  void main()
4.  {
5.      cout << get_sum( 5 , 4 );
6.  }
7.
8.  int get_sum(int a, int b)
9.  {
10.     return a + b;
11. }
```

التصريح عن الدالة

طباعة الدالة "ناتجها"

بناء الدالة

تطبيق المثال[1]:

```
#include <iostream.h>

int get_sum( int , int );
//////////////////////////////////
int main(){

    cout << get_sum()
};          int get_sum ( int a=0, int b=0)
//////////////////////////////////
int get_sum( int a=0, int b=0){
    return a + b;
}
```

الواجب:

- قم بعمل مثلث باسكال يقوم برسم عدة أشكال للمثلث باستخدام دالة واحدة تحوي دالة for واحدة، حيث يتم تغيير شكل المثلث بالتلاعب بقيمة متغيرات الدالة فقط.

- حل التمارين الموجودة لدى المصور إلى نهاية الترم.

(1) تم استخدام برنامج (Bloodshed Dev-C++) في هذا المثال وبعض الأمثلة في هذا الكتاب.

القيم الابتدائية في الدوال:

تعتبر كنوع من تحديد القيمة الابتدائية لمتغيرات الدالة أو الإجراء، فيمكنك عدم إسناد أي قيمة لمتغيرات الدالة:

مثال :

```
1.  double _div( float a=0, float b=0){
2.  {
3.     if(b==0){
4.        return 0;
5.     }else{
6.        return (a / b);
7.     }
8.  }
9.
10. void main()
11. {
12.     cout << _div() << endl;
13.     cout << _div(5) << endl;
14.     cout << _div(5, 0) << endl;
15.     cout << _div(8, 5) << endl;
16. }
```

تحديد قيم افتراضية لمتغيرات الدالة "أصفار"

إرجاع 0 إذا كان المتغير الثاني صفراً

إرجاع ناتج قسمة العددين إذا كان المتغير الثاني لا يساوي الصفر.

الناتج: 0
الناتج: 0
الناتج: 0
الناتج: 1.6

ملاحظات:

- يمكن جعل إحدى متغيرات الدالة تحمل قيمة ابتدائية والأخرى لا تحمل قيمة ابتدائية فهذا يعني أنه يجب إسناد قيمة لهذا للمتغير الآخر بشرط أن لا تجعل المتغيرات التي تحمل قيمة ابتدائية قبل المتغيرات التي لا تحمل قيم ابتدائية (لماذا؟ "ابحث عن هذا الموضوع").
- تم كتابة div_ تسبقها شرطة لأن هناك دالة من دوال مكتبة iostream.h تحمل الاسم div.

استخدام #Define:

يستخدم هذا الأمر ليخبر المترجم باستبدال سلسلة من الأحرف بالقيمة المجاورة للأمر define فهذا الأمر لا يفحص نوع القيمة فقد تكون قيمة أو معالجة لعملية حسابية أو غيرها كما في الدوال:

١) استخدام الأمر define لتعريف الثوابت:

الشكل العام Public formula:

```
#define Constant  value;
```

مثال :

```
1.    #define MAX  100;
2.    main ( )
3.    {
4.        cout << MAX;
5.    }
```

الناتج: 100

٢) استخدام الأمر define بدلا عن الدوال:

الشكل العام Public formula:

```
#define Function_name (parameters)   Statements...
```

مثال :

```
1.    #define SUM( x , y ) x + y;
2.    main ( )
3.    {
4.        Int z = SUM( 1 , 2 );
5.        cout << z << endl;
6.        cout << SUM(3.5, 7.5);
7.    }
```

الناتج: 3
الناتج: 11

مميزات define:

- لا يحتاج لتعريف نوع البيانات.
- لا يحتاج لتعريف نوع الدوال.
- يمكن إسناد قيمتها إلى متغير بشرط أن يكون المتغير من نفس نوع البيانات المعادة من الدالة.
- يمكن الاستغناء بها عن التحميل الزائد للدوال overload "سيتم دراسته في الفصل الثاني".

عيوب define:

- لا يمكن عمل مجموعة إجراءات "جمل" في سطور متعددة تحت الأمر define لأن المترجم سيتجاهل السطور اللاحقة ويعتبرها خطأً.

ملاحظات:

- نلاحظ أن هذه الطريقة لا تحتاج لتعريف نوع المتغيرات في الدالة SUM ، ويمكننا عند الاستدعاء أن نكتب أي قيم من أي نوع، لكن ما يحدد نوع المتغيرات المرسلة هو نوع العملية وهي "x+y" ففي هذه الحالة سيحدث خطأ عند إرسال قيم حرفية نظراً لأن جملة الدالة تحتوي على جمع ، حيث لا يمكن جمع قيم نصية.
- هذه الطريقة تشبه إلى حد كبير تعريف الثوابت const حيث لا يمكن تغيير القيمة بعد تعريفها.
- يجب أن يتم وضع define قبل الدالة الرئيسية main "في منطقة التصاريح العامة".