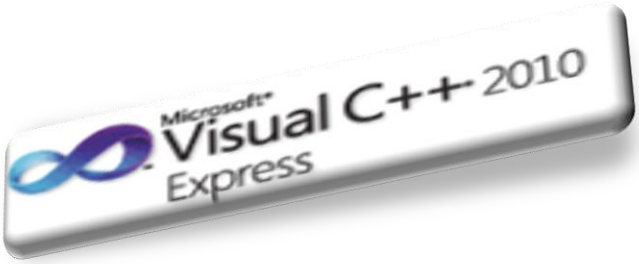
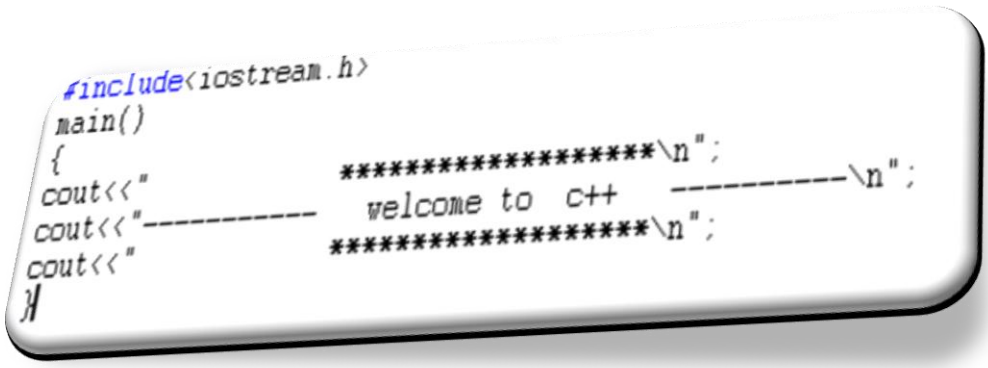
البرمجة بلغة C++





```
#include<iostream.h>
main()
{
cout<<"                    **********\n";
cout<<"----------    welcome to  c++   ----------\n";
cout<<"                    **********\n";
}
```

إعداد

أبومهاب

مقدمه

الحمد لله الذي هداني لهذا العمل وأسأله أن يجعله في ميزان حسناتي وأسأل كل من انتفع بهذا الكتاب أن يدعو لي بحسن الخاتمة وأن يدعو لأولادي أن يكونوا صالحين .

وجزاكم الله خيراً

أبومهاب

٢٠١٥

mohabalihassan@gmail.com

كتب أخرى لي :

١. الفيجول بيسك دوت نت

http://www.kutub.info/library/book/4500

٢. أكسس

http://www.kutub.info/library/book/7991

٣. صيانة الحاسبات والطابعات

http://www.kutub.info/library/book/18811

٤. نظم تشغيل الحاسبات

http://www.kutub.info/library/book/19143

٥. شبكات الحاسبات

http://www.kutub.info/library/book/19071

فهرس المحتويات

البـــــــــــاب الأول

البـــــاب الثـــــــاني

البـــاب الأول

يتكون الحاسب الآلي من مجموعة من الدوائر الإليكترونية وهو جهاز لا يفكر ولا يخطط لنفسه وبالتالي كانت الحاجة لظهور كوادر بشرية متخصصة تقوم بوضع البرامج التي تُخطر الوحدات المختلفة بخطوات تنفيذها للأوامر الواحد تلو الآخر،هذه الكوادر تسمى مبرمجين Programmers.

لغات البرمجة :

يستطيع من خلالها المبرمج كتابة مجموعة من الأوامر البرمجية بطريقة يفهمها الحاسب لتنفيذ مهام معينة ،ويوجد أنواع مختلفة من لغات البرمجة لكل منها قواعد وأوامر وطريقة كتابة كود تختلف عن لغة برمجة أخرى ،ويمكن أن يتقن المبرمج أكثر من لغة برمجة،وتختلف قدرات لغات البرمجة فمنها ما يُكتب به نُظم التشغيل وبالتالي تكون لها قدرة على التعامل مع امكانيات الذاكرة والهارد وير.

أقسام لغات البرمجة:

١. **لغات منخفظة المستوى** Low Level Languages

وهي لغات مرتبطة بتصميم الحاسب وتشتمل على:

أ -**لغة الآلة** Machine Languages

وهي اللغة التي يفهمها الحاسب ويتم تنفيذها مباشرة دون الحاجة إلى مترجم وهي النظام الثنائي**(0,1)** وهي لغة معقدة .

ب -**لغة التجميع** Assembly Languages

وفيها تم استبدال الرموز الرقمية في لغة الآلة بمجموعة من الكلمات الرمزية "المختصرة "، إذ من السهل التعامل معها، مثل: **L for Load, A for Add**،ويتم الإستعانة ببرنامج يسمى الـــمُجمع **Assembler** لتحويل البرنامج المكتوب بلغة التجميع إلى برنامج مكتوب بلغة الآلة،وتُستخدم لغة التجميع في برمجة المعالجات الدقيقة **Microprocessors** والدوائر المتكاملة **IC**.

*مثال(MOV AL, #61h) تعني نقل **Move** القيمة **61h** والتي تكافىء بالنظام العشري ٩٧ حيث **h** تعني **Hexadecimal** إلى "سجل المعالج" **Register** ذو الاسم **AL**، الأمر "**mov**" يكتب بكود التشغيل بالشكل ١٠١١ .*

٢. **لغات عالية المستوى** High Level Languages

وهي لغات قريبة في تركيبها من اللغات الطبيعية(مثل الإنجليزية)، ومنها لغة الفيجول بيسك ولغة السي،وتحتاج البرامج المكتوبة بلغات عالية المستوى إلى ترجمة للغة الآلة عن طريق المترجم Compiler ويتم عن طريقه التأكد من صحة البرنامج منطقياً ونحوياً.

شكل توضيحي لكيفية التحويل من اللغة عالية المستوى إلى لغة الآله

أنواع اللغات عالية المستوى:

أ - **لغات عامة الغرض:** مثل لغة **VB** ، **C**.

ب - **لغات التطبيقات التجارية :**مثل لغة **COBOL**

ج - **لغات التطبيقات الرياضية :**مثل لغة **Fortran**

د - **لغات استفسارية:** مثل لغة **SQL**

ه - **لغات الذكاء الإصطناعي :**مثل لغة **PROLOG**

و - **لغات البرمجة الشيئية :**مثل لغة **++C**و **C#.net**

البرنامج :

هو مجموعة من الأوامر البرمجية المكتوبه بأحد لغات البرمجة والتي يتم ترجمتها إلى لغة الآله،ولكل لغة برمجة أسلوبها الخاص في التعبير عن هذه الأوامر.

تصنيفات لغات البرمجة:

أ – **اللغات الإجرائية**Procedural Languages :

ويتكون فيها البرنامج من مجموعة من الخطوات والإجراءات المرتبة ترتيب منطقي لتحقيق نتائج معينة ويمكن الوصول لأي مكان في البرنامج عن طريق جمل GOTO مثل لغة البيسك ،وهذه اللغات تحتاج مترجم لتحويلها إلى لغة الآلة.

مثـــال

```
10 Read A, B
20 C=A*B
30 Print A;"*"; B;"="; C
40 Goto 10
50 Data 6, 4, 2,3,5,7
60 End
```

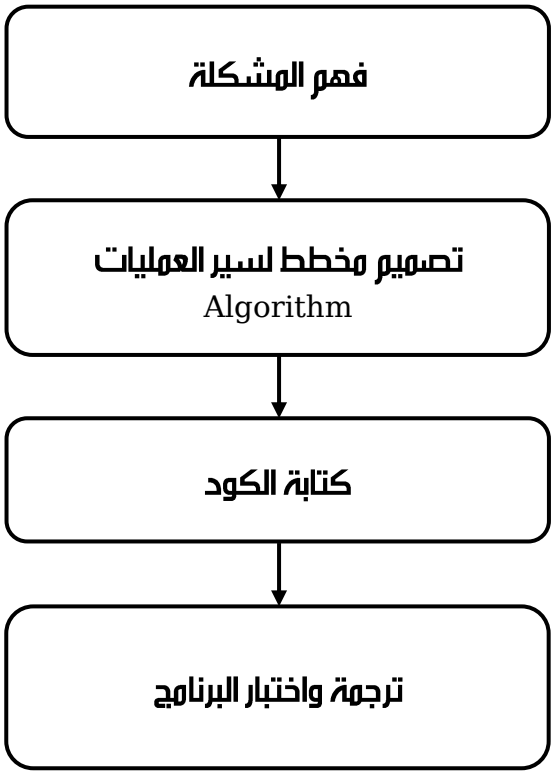
ب - **اللغات الهيكلية** Structural Languages:

وفيها تم استخدام الدوال والتي تتضمن أكثر من أمربرمجي والتي يمكن استدعائها بسهولة عن طريق اسم الدالة وكذلك جمل التحكم وجمل التكرارات مثل لغة الباسكال ولغة السي.

ج - **لغات البرمجة الشيئية(كائنية التوجه)** Object Oriented Languages:

وهي لغات تقوم بفصل البرنامج وظيفياً وشكلياً إلى كائنات Objects مثل النوافذ وصناديق الحوار والتي تعمل باستقلال تام،كل منها له مواصفات (حجمه ولونه)تحدد شكله وسلوكه Properties وله أحداث Events تقع عليه مثل لغة C++ ولغة الـــ JAVA.

مراحل البرمجة



شكل توضيحي لمراحل البرمجة

نبذه عن لغة السي :

ظهرت لغة السي في السبعينات عن طريق كين تومسن ودينيس ريتشي واستخدمت لبرمجة نظام التشغيل يونكس وهي مشتقة من لغة تسمى B ولذلك سميت بالحرف التالي في اللغة الإنجليزية وهو الـــ C،والتي خرجت منها إصدارات مثل C++ و C++.Net و C#وC#.Netوهي من لغات البرمجة الشيئية والتي تعتبر من الجيل الرابع للغات البرمجة.

دنيس ريتشي كين تومسن

مميزات لغة C++ :

١. لغة عامة الغرض: فهي بيئة تطويرية كاملة لتطوير البرامج مثل برامج قواعد البيانات والرسومات والحسابات ونظم التشغيل
٢. لغة تركيبية:حيث يتألف البرنامج المكتوب من دالة رئيسية وبداخلها مجموعة من الدوال الإجرائية، وكل دالة من هذه الدوال عبارة عن مجموعة من الأوامر.
٣. لغة متنقلة:يمكن للبرنامج المكتوب بهذه اللغة أن يعمل مع أكثر من جهاز ومع أنظمة تشغيل مختلفة.
٤. لغة قياسية: معظم مترجمات اللغة تتوافق مع مترجمات اللغات القياسية الأخرى.

مميزات لغة C++ عن لغة السي :

١. تعريف المتغيرات وقت الحاجة إليها وفي أي مكان حيث كان يشترط في لغة السي الإعلان عن المتغير في بداية البرنامج.
٢. سهولة كتابة التعليقات بعد العلامة "//"ودون التقيد بعلامة في نهاية التعليق كما في لغة السي حيث كان التعليق يبدأ وينتهي بـــ*/
٣. استدعاء الدوال بأكثر من طريقة
٤. تدعم أسلوب البرمجة الشيئية مما يسهل إنشاء ما يسمى بالـــ Class وتوريثه.

رموز لغة C++:

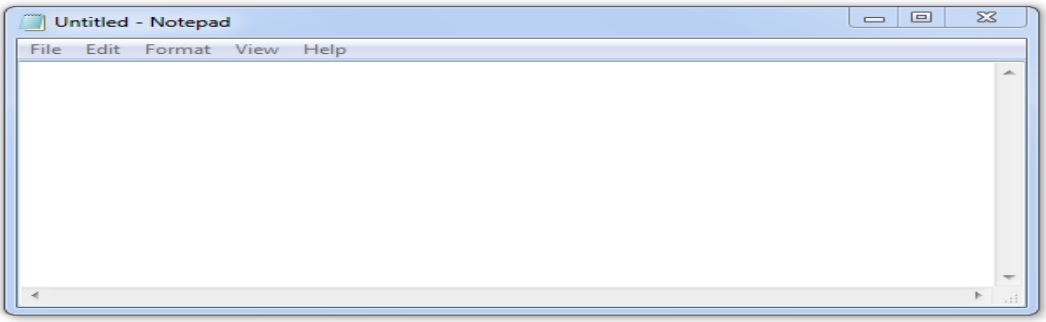
١. الحروف الإنجليزية الكبيرة...A,B,C,
٢. الحروف الإنجليزية الصغيرة ...a,b,c,
٣. الأرقام ...1,2,3,
٤. الرموز الخاصة(.....,”,;,=!,[],!,||,(),%,#,$,>,<,//,/,-,+)

ملاحظة:لغة C++ حساسة الأحرف فـــ Ali تختلف عن ali

الأدوات اللازمة لبناء برنامج بلغة C++:

اولاً: محرر نصوص Text Editor:

يتم كتابة البرنامج في لغة C++ على أي محرر نصوص مثل برنامج Note Pad الموجود على نظام التشغيل ويندوز.



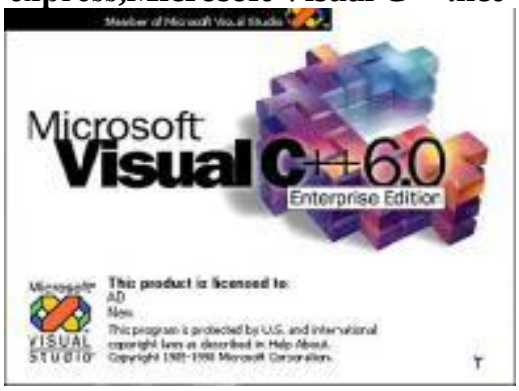
ثانياً مترجم Complier:

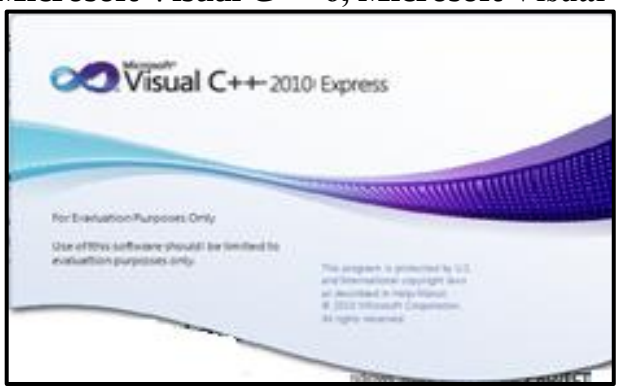
تقوم المترجمات بترجمة الملفات المصدرية إلى لغة الآلة إذا لم تكن هناك أخطاء في قواعد اللغة وبعد الترجمة يتم إنشاء الملف بصيغة Obj.

وجميع المترجمات الحديثة بها بيئة تطوير متكاملة IDE .

امثلة للمترجمات :

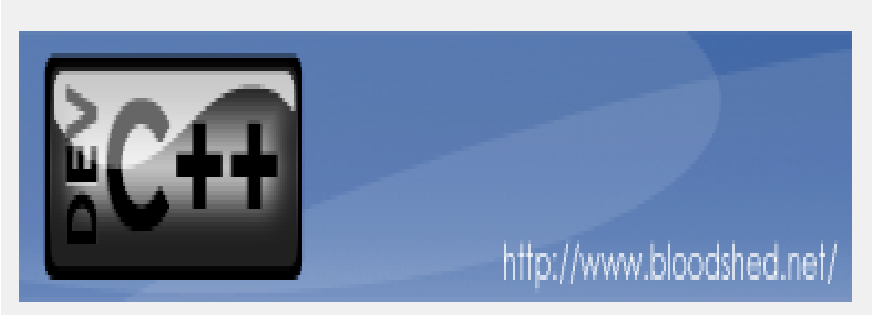
١. **Microsoft Visual C++ 6, Microsoft Visual C++ 2010 express,Microsoft Visual C++.net**



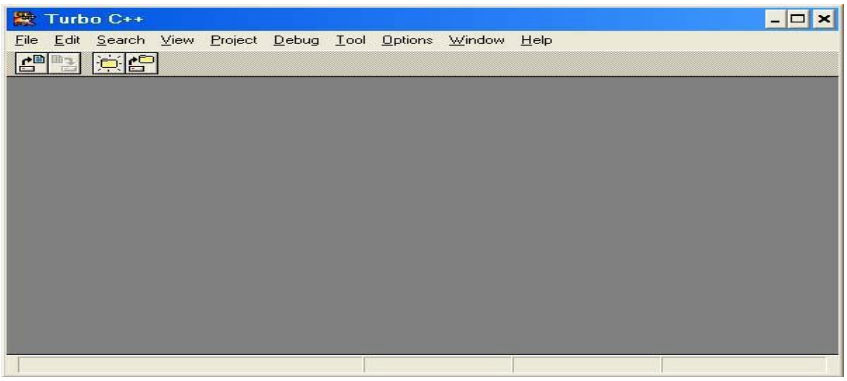


يجب تثبيت أداة **Microsoft Visual C++ Redistributable** وهي منصة البرمجيات التي تسمح لك بتشغيل المحتوى المكتوب بلغة البرمجة **C++** للتشغيل الصحيح للبرمجيات والألعاب.

٢. **Dev-C++**



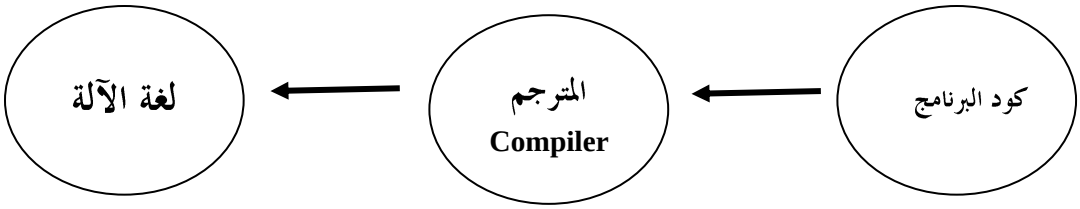
٣. **Turbo C++**



٤. **Borland C++**

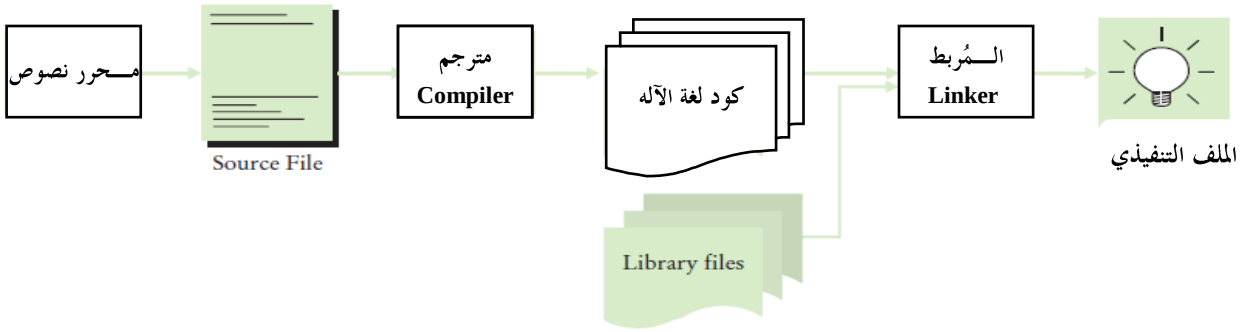
لماذا نحتاج إلى مترجم:

لأن الحاسب لا يستطيع أن يفهم سوى لغة الآلة وبالتالي نحتاج لمترجم ليحول الملف من اللغة عالية المستوى التي يفهمها المبرمج إلى لغة الآله التي يفهمها الحاسب و يتم خلال عملية الترجمة اكتشاف الأخطاء النصية والمنطقية في البرنامج.



ثالثاً المُربط Linker

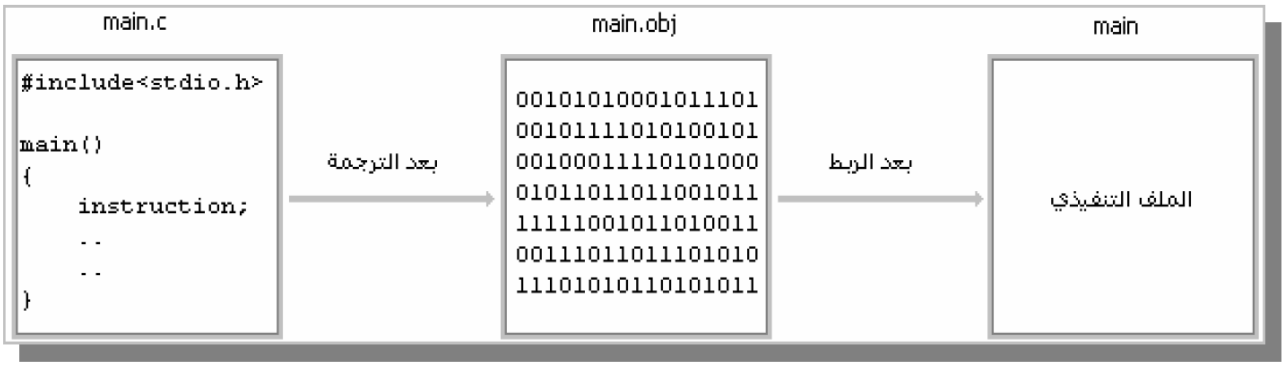
يقوم بتجميع الملفات ذات الصيغة obj. ثم يعطينا البرامج التنفيذية والتي تكون غالباً بالإمتداد exe. أو ملفات الربط الديناميكي DLL.



شكل توضيحي لخطوات بناء البرنامج في لغة C++

خطوات تنفيذ البرنامج في لغة C++:

١. كتابة البرنامج وحفظه باستخدام أحد برامج التحرير مثل Note Pad ثم نقوم بحفظ الكود بالإمتداد cpp.
٢. ترجمة البرنامج Compiling عن طريق أحد المترجمات مثل VC++ وينتج عن عملية الترجمة الصيغة obj.
٣. عملية الربط بمكتبات اللغة Linking وينتج عنها النسخة التنفيذية exe. أو نسخة DLL.



مرحلة إنشاء ملف تنفيذي

الأخطاء التي تصاحب تنفيذ البرنامج:

١. أخطاء المترجم Compiler Error:وهي تحدث نتيجة خطأ في قواعد كتابة البرنامج مثل عدم كتابة فاصلة منقوطه.
٢. أخطاء الربط Linking Errors:وتحدث عندما لا يتمكن الرابط Linker من إيجاد بعض الدوال أو عناصر البرنامج.
٣. أخطاء التنفيذ Run time Errors:وهي أخطاء تظهر أثناء تنفيذ البرنامج مثل القسمة على صفر.
٤. أخطاء نحوية Syntax Errors:وهي أخطاء في عملية الكتابة وتسبب نتائج خاطئة.

أنواع البيانات الأساسية في لغة الـ C++:

أولاً :الثوابتConstants

وهي عبارة عن قيم ثابتة لا تتغير طوال تنفيذ البرنامج وتنقسم إلى ثوابت عدديةوثوابت رمزية .

١. الثوابت العددية:وتشتمل على:

أ -الثابت العددي الصحيح Integer:مثل **15** و**20-** و**5,500+**

ب -الثابت العددي الحقيقي Float:مثل **12,5**و**788,2+** و**454,6-**

٢. الثوابت الرمزية :

وهى عبارة عن رموز اللغة وتتكون من الحروف والأرقام وتكون بين علامتى تنصيص مثل:

("Ali" - "TT" – "123" – "80+20 ")

كيفية الإعلان عن الثابت

تبدأ الجملة بكلمة Const ثم يذكر نوع المتغير ثم اسمه بحيث يفصل بينهما فراغ ثم علامة = ثم قيمة الثابت وتنتهي الجملة بفاصله منقوطة.

مثال:

```
Const Float Pi =3.14 ;
```

طريقة أخرى لتعريف الثابت باستخدام #Define

الشكل العام Public formula:

```
#define Constant  value;
```

مثال :

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

الناتج: 100

ثانياً :المتغيرات Variables

وهو عبارة عن أسماء تحجز مواقع فى الذاكرةRAMحتى يتمكن البرنامج من تخزين البيانات فيها وهذه المواقع تتغير أثناء تنفيذ البرنامج.

ملاحظات:

- المتغيرات تختلف في الحجم الذي تشغله في الذاكرة حسب نوعها فذاكرة الحاسب مقسمة إلى Bytes حيث نستطيع تخزين حرف واحد أو رقم طوله ٨ بت في البايت الواحد(من ٠ إلى ٢٥٥) .
- يمكن استعمال المتغيرات في أي مكان في البرنامج لكن يجب تعريفها قبل استعمالها.
- يمكن تعريف المتغيرات التي تنتمي إلى نفس النوع في سطر واحد،فقط نحتاج أن نُعلم المترجم في بداية البرنامج عن أنواع المتغيرات التي نريد استخدامها.

وتنقسم أنواع المتغيرات إلى:

١. متغير حرفي Char ويحتل واحد بايت في الذاكرة لتخزين الحروف والعلامات والأقواس والأرقام (يتعامل معها على أنها حروف) ولا تُجرى عليها عمليات حسابية أو منطقية .مثال.

```
Char a ,b;
a='ali',b='Ahmed'
```

٢. متغير صحيح Int

ويحتل ٢ بايت في الذاكرة في أنظمة ١٦ بت لتخزين أرقام بين 32767 إلى –32768 ويطلق عليه Short

مثال: ShortInt X;

بينما يحتل ٤ بايت في أنظمة ٣٢ بت لتخزين أرقام بين 2147483647 إلى –2147483648 ويطلق عليه Long

مثال: longInt X;

٣. متغير حقيقي صغير Float(عشري) (٧ أرقام)وهي أرقام تحتوي على فاصلة عشرية وتحتل ٤ بايت في الذاكرة.

مثال : Float y;

٤. متغير حقيقي طويل Double(عشري)(١٥ رقم)وهي أرقام حقيقية طويلة تحتوي على فاصلة عشرية وتحتل ٨ بايت في الذاكرة.مثال: Double Z;

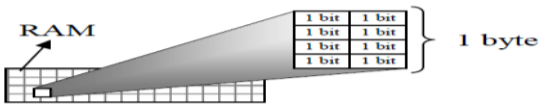
٥. متغير حقيقي طويل جدا Long Double(عشري) وهي أرقام حقيقية طويلة جدا تحتوي على فاصلة عشرية وتحتل 10 بايت في الذاكرة.مثال: Longdouble Z;

٦. متغير Bool (منطقي) له قيمتين إما True or False

ملاحظة :يدل التعبير Signed على الأرقام السالبة بينما Unsigned يدل على الأرقام الموجبة وإذا لم يذكر أي من التعبيرين فيدل على أنه Signed.

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

الجدول التالي يوضح أمثلة :

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(ضعف **Float**)	7,533,039,395
Long double	أرقام حقيقية ضخمة	9,176,321,236,01202,6
Unsigned Short Int	أرقام صحيحة قصيرة موجبة فقط	من 0 إلى 65535
signed Short Intأو Short Int	أرقام صحيحة قصيرة موجبة وسالبة	من 32767 إلى -32768

قواعد تسمية المتغير :

١. يجب ألا يبدأ برقم
٢. يجب ألا يحتوي على فراغات
٣. يجب ألا يحتوي أي علامات خاصة مثل +,*,@,#
٤. يجب ألا يحتوي أي من الكلمات المحجوزة مثل If ,Class ,For ,Void,while
٥. يمكن إضافة الأرقام لإسم المتغير مثل Ali6Jk
٦. يراعى حالة الأحرف فالمتغير ALI يختلف عن المتغير ali
٧. عند استخدام اسم متغير مكون من كلمتين نضع بينهما شرطة تحتية Ali_Hassan
٨. بعض المترجمات لا تقبل طول متغير أكبر من ٣٢ حرف حيث تهمل باقي الأحرف

الكلمات المحجوزة هي :

```
auto break case catch char class const continue default
delete do double else enum extern float for friend
goto if int inline long new operator private protected
public register return short signed sizeof static struct
switch template this throw typdef union unsigned virtual
void volatile while
```

أمثلة صحيحة لأسماء المتغيرات

Ali Door_4 X_ray B6

أمثلة خاطئة لأسماء المتغيرات

7ahg **السبب**:بدأ بحرف

B8#7 **السبب**:لاستعماله الرمز #

jتT **السبب** :استعماله حروف غير انجليزية

Do **السبب**:كلمة محجوزه

كيفية الإعلان عن المتغير:

يذكر نوع المتغير ثم اسمه بحيث يفصل بينهما فراغ وتنتهي الجملة بفاصله منقوطة.

وبالتالي فالصيغة العامة لتعريف المتغير هي: نوع المتغير اسم المتغير ;

مثال على تعريف متغير حرفي: Char a;

مدى عمل المتغيرات:

١. المتغيرات المحلية Local Variable:

وهذه المتغيرات تُعرف داخل جسم الدالة ولا تستمر فعاليتها بعد انتهاء تنفيذ الدالة أي بعد إعادة القيمة من الدالة.

٢. المتغيرات العامة Global Variables :

وهذه المتغيرات تُعرف خارج جميع الدوال فلها تأثير عام على كامل البرنامج بكل دواله وهي قليلة الإستخدام.

الفرق بين الثابت والمتغير

	الثابت	المتغير
الحيز التخزيني	نفس الحيز التخزيني للمتغير	نفس الحيز التخزيني للثابت
القيمة	غير قابله للتغير	قابله للتغير
طريقة الإعلان	يتم الإعلان عنه باستخدام الكلمة المحجوزة Const يعقُبها نوع الثابت ثم تعيين قيمة للثابت. مثال: Const Float Pi =3.14 ;	يذكر نوع المتغير ثم اسمه بحيث يفصل بينهما فراغ ويتم تعيين القيمة على يمين المتغير بعد علامة = مثال: Int a; a=5 ;

المعاملات (المشغلات) Operators

١. المعاملات الحسابية والمنطقية

المعامل	الوظيفة	العمليات المسموح بها	نوع البيانات المستخدمة فيها
+	الجمع	أدوات حسابية ومنطقية	عددية ومنطقية
−	الطرح	أدوات حسابية	عددية
/	القسمة	أدوات حسابية ومنطقية	عددية
%	باقي القسم	أدوات حسابية	عددية
*	الضرب	أدوات حسابية	عددية
++	الزيادة بمقدار واحد	أدوات حسابية	عددية
−−	النقصأن بمقدار واحد	أدوات حسابية	عددية
=	التخصيص	أدوات حسابية ومنطقية	عددية ومنطقية
>	الأكبر (اكبر من)	أدوات علاقية	عددية ومنطقية
<	الأصغر (أصغر من)	أدوات علاقية	عددية ومنطقية
<=	أصغر من او يساوى	أدوات علاقية	عددية ومنطقية
==	أكبر من أو يساوى	أدوات علاقية	عددية ومنطقية
!=	إذا كان لا يساوى	أدوات علاقية	عددية ومنطقية
&&	and	أدوات منطقية	منطقية
!!	or	أدوات منطقية	منطقية
!	Not	أدوات منطقية	منطقية

استخدام العمليات الحسابية الأساسية في C++:

العامل	الوظيفة	التعبير الجبري	التعبير في C++
+	جمع	B+h	B+h
-	طرح	B-h	B-h
*	ضرب	Bh	B*h
/	قسمة	B/h,	B/h
%	الباقي	B mod h	B%h

ملاحظة هامة:

قسمة عدد صحيح على عدد صحيح يكون الناتج صحيح	قسمة عدد حقيقى على عدد حقيقى يكون الناتج حقيقى
قسمة عدد حقيقى على عدد صحيح يكون الناتج حقيقى	لا توجد عملية قسمة عدد صحيح على عدد حقيقى

مشكلة الخلط بين = و== =

= هو معامل الإسناد مثال x=6 يعني إسناد قيمة **6** إلى **X**

== هو معامل علائقي مثال x==6 مثل استخدامه مع جملة IF لإختبار قيمة **x**

أولويات العمليات الحسابية Arithmetic Operations :

١ -الزيادة والنقصان عندما تأتى قبل العدد – – ،++

٢ -الأقواس ( )

٣ -إشارة السالب –

٤ -القسمة وباقى القسمة والضرب /، % ، * (تقدم العملية الأقرب لليسار)

٥ -الجمع والطرح + ، –

٦ -التساوى =

٧ -الزيادة والنقصان المتأخرة بعد العدد ++ ، --

أمثلة :

ما قيمة **X** في الأمثلة التالية:

أ.**X=4+2*5**

ب.**X=6/3*7+2**

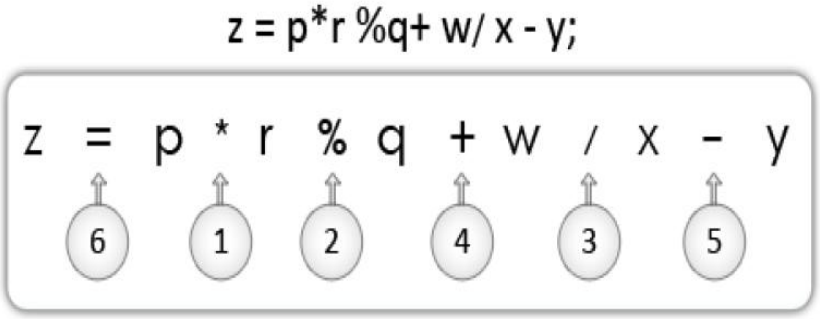
ج.**X=5+(3/3)**

الحل :

أ.**X=4+10=14**

ب.**X=2*7+2=14+2=16**

ج.**X=5+1=6**



٢. **معاملات المقارنة** Relation Operator

في++C توجدعمليات المقارنة حيث بإمكانك مقارنة قيمة مع قيمة فتكون النتيجة إما صحيحة True أو خاطئة False.

ويوجد ٦ عمليات مقارنة في **C++** هي:

الرمز	المعنى	مثال
==	يساوى	a==b
!=	لايساوى	a!=b
>	أكبر من	a>b
<	أصغر من	a<b
>=	أكبر من أو يساوى	a>=b
<=	أصغر من أو يساوى	a<=b

أمثلة :

5==3 تكون النتيجة False أو 0

5>3 تكون النتيجة True أو1

٣. المعاملات المنطقية Logical Operators

وهي تُستخدم في عمليات المقارنة ولها رموز خاصة وهي:

الرمز	التوضيح
&&	و
||	أو
!	نفي

أمثلة:

```
5<3 && 5>3);
5==3 && 3==5);
5>3 && 5<3);
5<3 || 5>3);
5==3 || 3==5);
5>3 || 5<3);
!(5<3));
!(5==3));
!(5>3));
```

٠ : لأنه توجد علاقة خاطئة و هي 5<3.
٠ : لأن كلا العلاقتين خطاءتين.
٠ : لأنه توجد علاقة خاطئة و هي 5<3.
١ : لأنه توجد علاقة صحيحة و هي 5>3.
٠ : لأن كلا العلاقتين خطاءتين.
١ : لأنه توجد العلاقة صحيحة و هي 5>3.
١ : لأن ٥ ليست أصغر من ٣.
١ : لأن ٥ لا تساوي ٣.
٠ : لأن ٥ أكبر من ٣.

٤. معامل الزيادة increment Operator (++)

حيث تتم زيادة المتغير بقيمة واحد وقد يكتب المعامل++ قبل اسم المتغير بحيث تتم الزيادة قبل تنفيذ الأمر الحالي أو يكتب المعامل ++ بعد اسم المتغير بحيث تتم الزيادة بعد تنفيذ الأمر الحالي مباشرة.

```
int a=5,b=8;
cout<<"\na= "<<++a;
cout<<"\nb= "<<b++;
cout<<"\n after incrementation b= "<<b;
}
```

الناتج

```
a=6
b=8
after incrimination b=9
```

a ++ تكافىء a=a+1

٥. معامل الإنقاص Decrement Operator(--)

حيث يتم إنقاص المتغير بقيمة واحد وقد يكتب المعامل-- قبل اسم المتغير بحيث يتم الإنقاص قبل تنفيذ الأمر الحالي أو يكتب المعامل – – بعد اسم المتغير بحيث يتم الإنقاص بعد تنفيذ الأمر الحالي مباشرة.

أداة تعيين الطول size of

تستخدم لتعيين طول المتغيرات بالبايت

```
Float x;
Z=size of (x);
```

نتيجة Z هي ٤ بايت وهو طول X

الفاصلة comma

تفصل بين تعبيرين

مثال Int x, y;

الشكل العام للبرنامج في لغة C++

رأس البرنامج Header Files

```
1.  #include <library_name.h>
2.  Public  Declaration
3.  Main ()
4.  {
5.          Private Declaration
جسم البرنامج Program Body
6.          Statements..
            Statements..
            Statements..
7.  }
```

١. استيراد المكتبات

٢. منطقة التصاريح العامة

٣. الدالة الرئيسية

٤. بداية الدالة الرئيسية

٥. منطقة التصاريح الخاصة

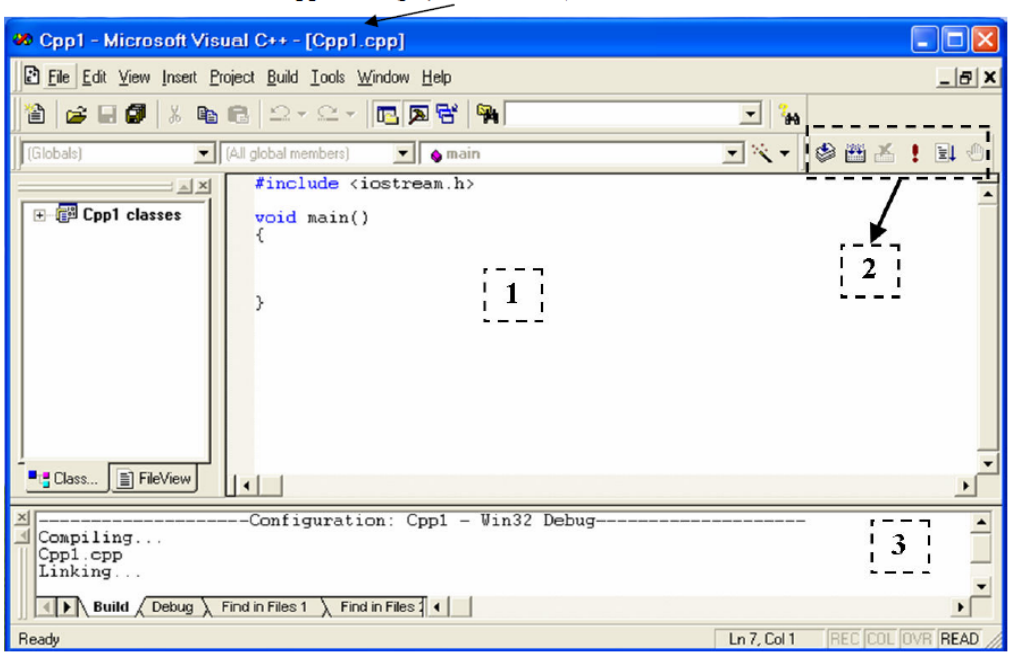
٦. جمل برمجية

٧. نهاية الدالة الرئيسية

الواجهة الإفتتاحية لفيجول ستوديو٦

واجهة البرنامج:

اسم الملف المصدري مع الامتداد cpp.



١. مكان كتابة الكود

٢. تنفيذ البرنامج

٣. منطقة عرض الأخطاء

كتابة البرنامج الأول :

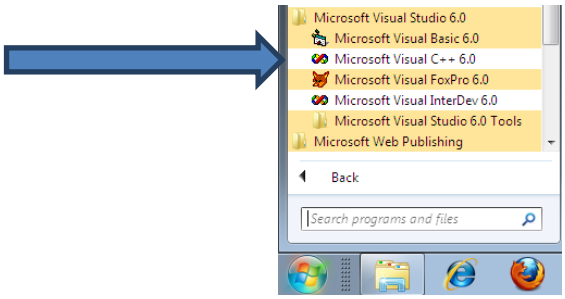
برنامج يُظهر رسالة ترحيب على الشاشة

```
#Include <iostream.h>
main ()
{
Cout<<"Welcom to C++ ! \n";
return (0) ;
}
```

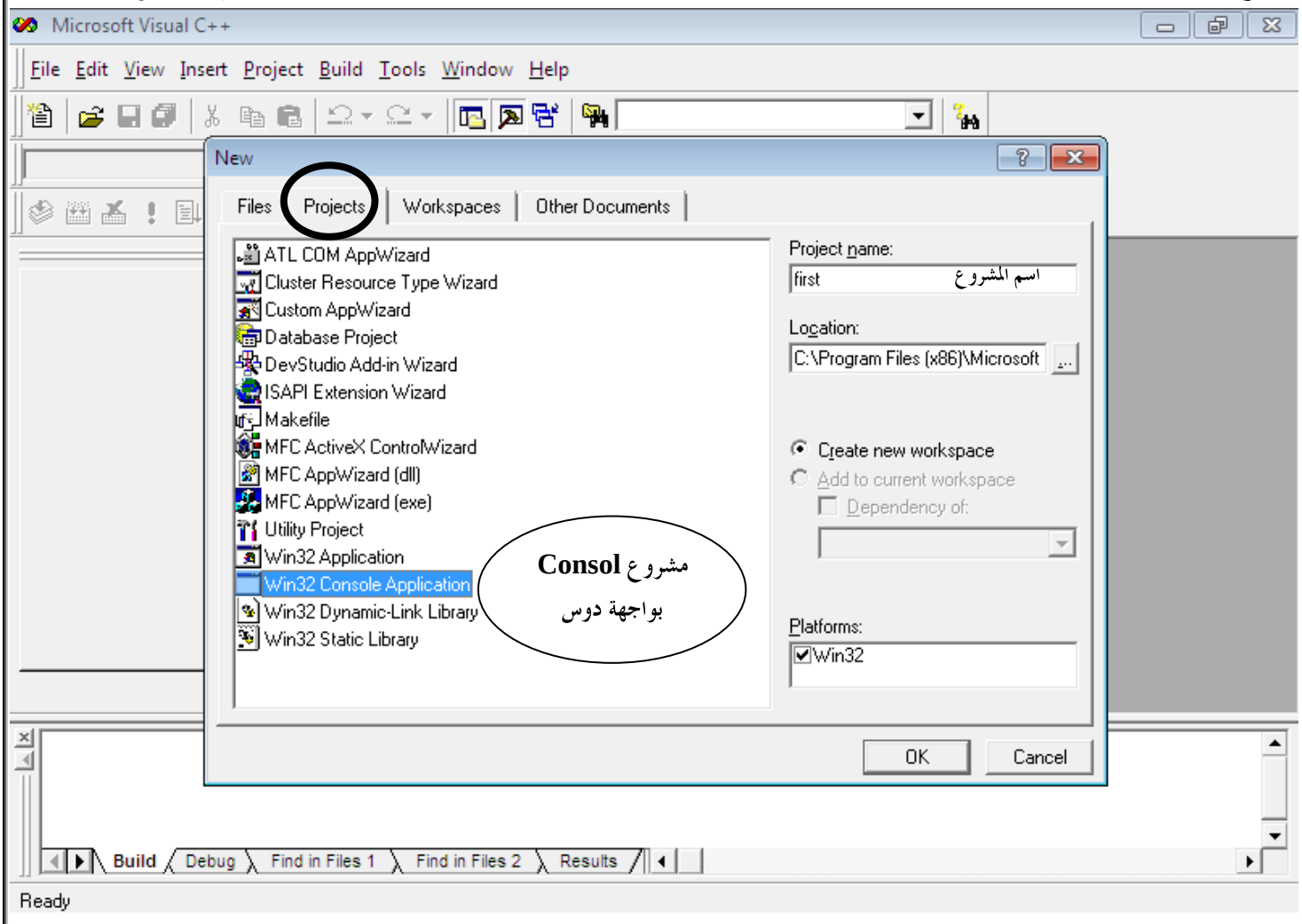
افتح visual c++ **واتبع الخطوات التالية لكتابة البرنامج :**

١. إنشأ مجلد لحفظ المشاريع مثلا c:\c++

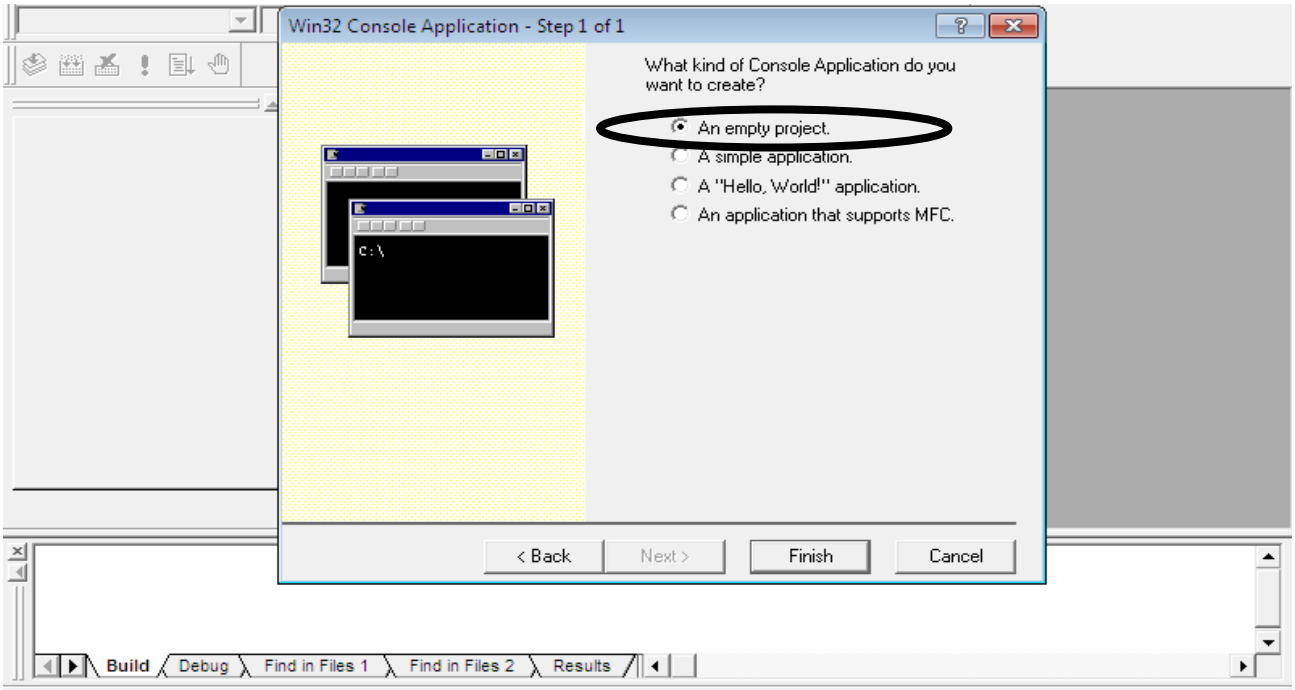
٢. افتح Microsoft Visual Studio واختر Microsoft Visual C++ 6.0



٣. افتح قائمة File–New واختر من تبويب projects الإختيار Win32 Consol Application وحدد اسم المشروع

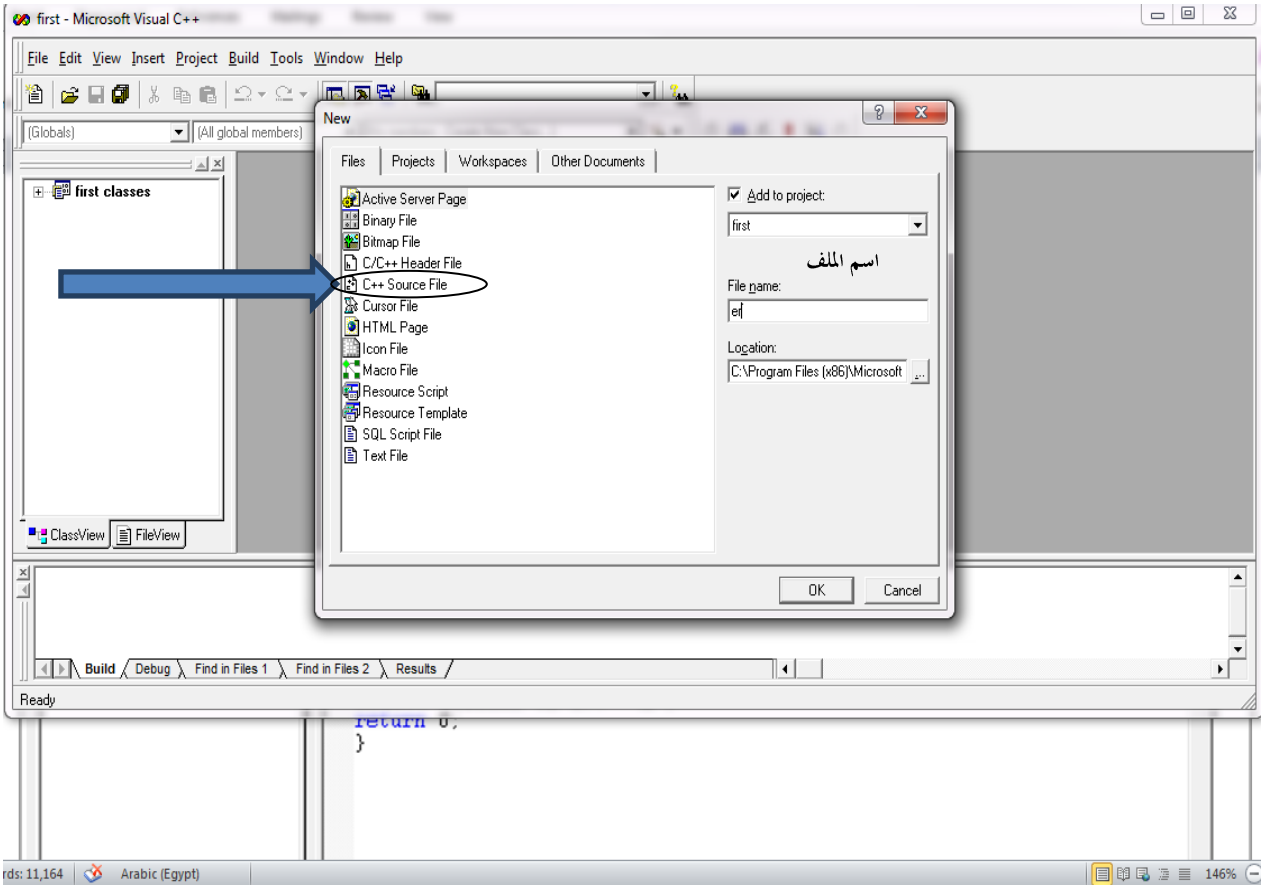


٤. اختر An Empty Project ثم اضغط زر Finish

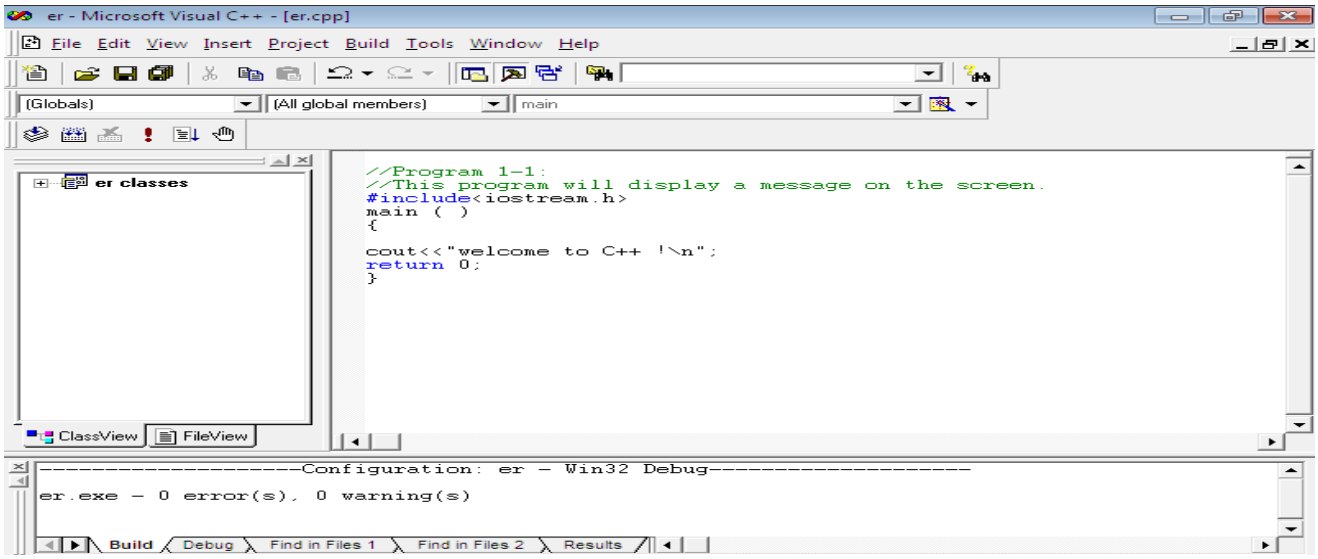


٥. لكتابة المشروع يمكن استخدام المحرر الموجود مع لغة Visual C++ 6.0 ولفتحه اختر
File –New –C++ Source File

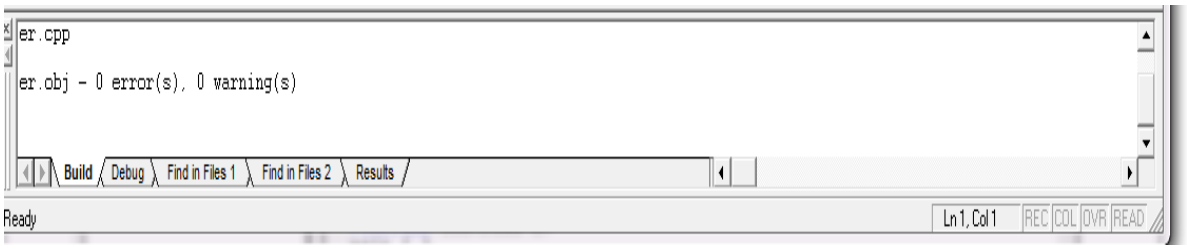
٦. ضع اسم للملف



٧. اكتب البرنامج في منطقة كتابة الكود



٨. لترجمة البرنامج افتح قائمة Build ثم Compile ويظهر الشكل التالي في أسفل شاشة البرنامج وهو الجزء الخاص بعرض أخطاء الترجمة.



٩. ولعمل النسخة التنفيذية من المشروع افتح قائمة Build ثم Build First.exe أو اضغط مفتاح F7

١٠. استخدم الأمر Ctrl+F5 لتنفيذ البرنامج وإظهار شاشة المخرجات .

١١. إذا لم يتم إنشاء النسخة التنفيذية لوجود أخطاء اضغط مفتاح F4 لمعرفة السطور التي بها أخطاء .

١٢. لتشغيل النسخة التنفيذية توجه إلى مسار حفظ المشروع وستجد ملف التشغيل داخل مجلد Debug

شرح أجزاء البرنامج :

١. السطرين المكتوبين باللون الأخضر والمسبوقين بعلامة \\هما تعليق وهما يوضحان وظيفة البرنامج ولا يشترط كتابة تعليق التعليقات على البرنامج.

الطريقة الأولى:هى كتابةالتعليق بعد العلامة (//) حيث يتجاهل المترجم السطر الذي يلي هذه العلامة.

```
// this is a;
```

ولكن لوتجاوز التعليق السطر لزم إضافةالمزيد من الرموز (//) أمام كل سطر من التعليقات

الطريقة الثانية: وهى كتابةالتعليق بين العلامتين (/*) بدايةالتعليق و(*/) نهايةالتعليق ومهما كان عدد أسطر التعليق فلن يُلتفت المترجم إلى هذه السطور.

```
/* This is the second method*/
```

٢. #<iostream.h> تكتب في بداية البرنامج فالرمز # يقوم بتوجيه المترجم لإستخدام مكتبة iostream وهي مكتبة دوال الإدخال والإخراج وبدونها لا يمكن إدخال قيم للبرنامج أو طباعة على الشاشة فهذه المكتبة تشمل أمر الإدخال cin وأمر الإخراج Cout.

٣. الدالة ( ) Main

وهي النقطة الأولى لبداية أي برنامج بلغة C++ وبدونه لا يوجد برنامج حيث يبدأ تنفيذ الأوامر بعدها حتى لو كُتبت في وسط البرنامج والأقواس () تشير إلى أنها دالة .

٤. القوسين {}

ويتم كتابة أوامر البرنامج بينهما

٥. Cout<<"Welcome to c++!"

الدالة cout تُستخدم لطباعة المخرجات على الشاشة ويكتب بعدها الرمز <<

ويُستخدم مع الدالة cout بعض الرموز الخاصة لتنسيق المخرجات على الشاشة مثل :

\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

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

٦. ; Semi colon :تنتهي بها كل جملة من جمل البرنامج وبدونها يعطي البرنامج رسالة خطأ.

٧. Return (0) وهي تُخبر المترجم بنهاية البرنامج وأن البرنامج انتهى دون أخطاء أثناء التنفيذ.

فضاء الأسماء Name Space

هو تقسيم أوامر C++ إلى مجموعات فمثلاً لإستخدام أوامر الإدخال والإخراج Cin,Cout فإنك تحتاج لإخبار المترجم أنك تستخدم فضاء الأسماء القياسية std مثال:

```
#include<iostream>
Using name space std;
```

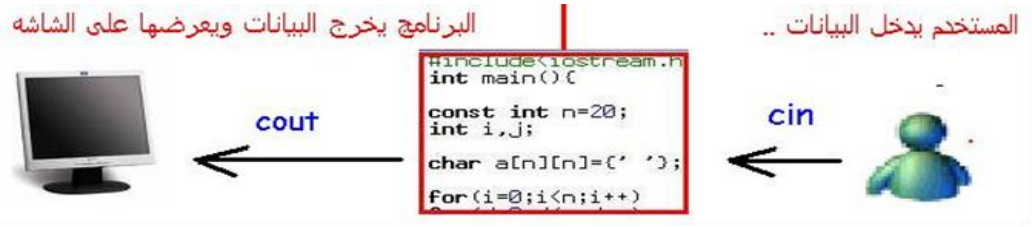
ولكن هناك مترجمات لا تفهم السطرين السابقين وإنما تفهم الشكل التالي:

```
# include<iostream.h>
```

أنواع المكتبات :

١. *iostream.h وهي مكتبة عامة لأوامر الإدخال والإخراج*

- وقد تم دمج مكتبتين في هذه المكتبة هما istream و ostream
- وتشمل مكتبة iostream.h على دوال مثل :
  - Cin لإدخال قيم للبرنامج مثل cin>>x;
  - Cout لإخراج القيمة على الشاشة مثل cout <<x;



٢. *stdio.h المكتبة القديمة لأوامر الإدخال والإخراج(مكتبة لغة C)*

وتشتمل مكتبة stdio.h على دالتين هما :

i. Printf وهي دالة خاصة بعملية الإخراج وتتميز عن cout أنها لا تحتاج إلى كتابة الرمز >>

مثال:

```
#include<stdio.h>

main()
{
     printf("Welcomt to c++ ");
return(0);
}
```

ii. Scanf وهي دالة خاصة بعملية الإدخال مثل دالة Cin

مثال: Scanf("%d",&x);

ملاحظة:

١.يجب كتابة الرمز & قبل اسم المتغير

٢.يشير الرمز %d أن المتغير x من نوع int

٣. *Conio.h مكتبة دوال أوامر الشاشة*

يقوم vc++ بتنفيذ البرنامج ويعود سريعاً للمحرر IDE ولكن إذا أردت أن تُثبت المخرجات على الشاشة فعليك إضافة التالي إلى أول البرنامج:

#include<conio.h>

وإضافة العبارة: getch( ) في السطر الذي يسبق العبارة return (0)

مثال:

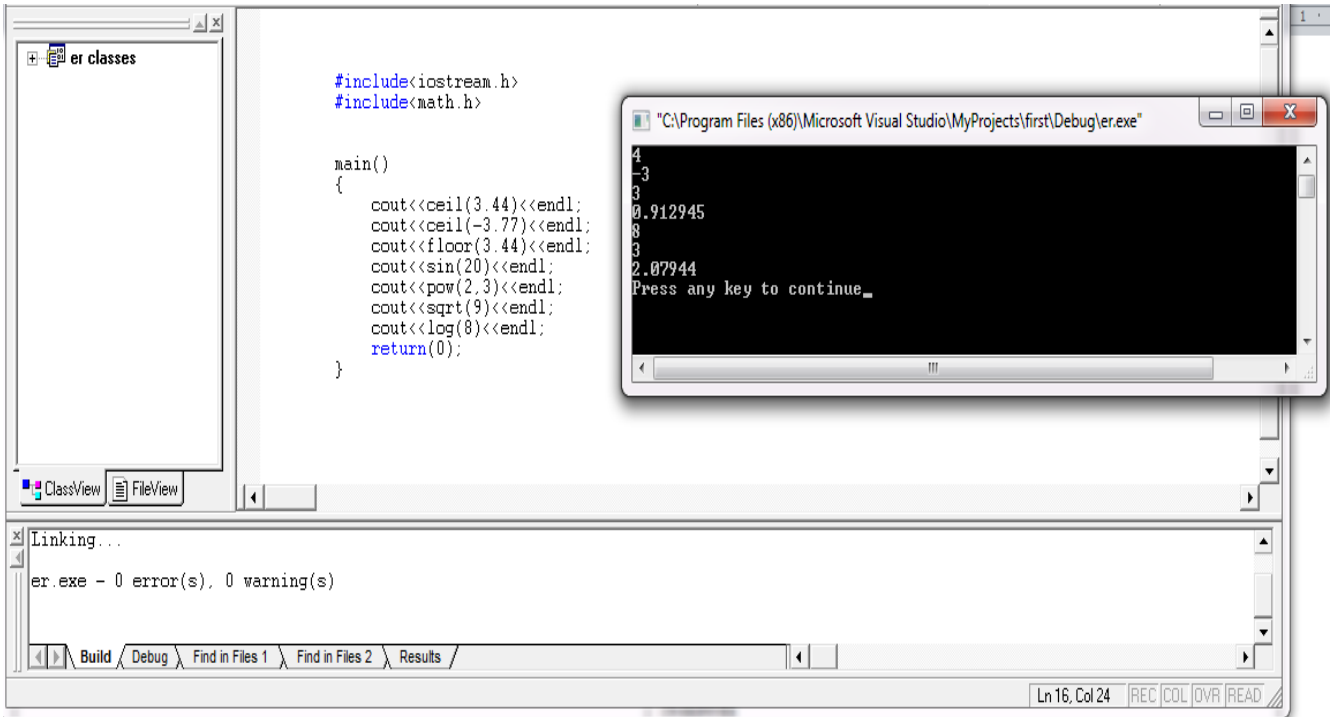
```
#include<iostream.h>
#include<Conio.h>

main()
{
     cout<<"Welcome to C++! \n";

     getch();
     return(0);
}
```

٤. Math.h مكتبة دوال الأوامر الرياضية

تحتوي على دوال رياضية كثيرة مثل:

الدالة	الرمز الرياضي	توضيح
abs(x)	| x |	الأعداد الحقيقية
sin (x)		جا س
cos (x)		جتا س
tan (x)		ضا س
sinh (x)		جا-١ س
cosh (x)		جتا-١ س
tanh (x)		ضا-١ س
pow (x , y)	x^y	الاس
exp (x)	e^x	e
sqrt (x)	$\sqrt{x}$	الجذر
log (x)	Log x	اللوغاريتم
ceil (x)		تقريب الكسور للأعلى
floor (x)		حذف الكسور

مثال:



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)

س١: كيف تكتب المعادلة التالية برمجياً

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

ج:

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

س٢: كيف تكتب المعادلة التالية برمجياً

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

ج:

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

٥. ***String.h*** *مكتبة دوال معالجة النصوص*

توفر هذه المكتبة نوع من أنواع البيانات وهو **String** الذي يقبل تخزين مجموعة حروف ورموز وأرقام كنص في متغير واحد.

وهذه المكتبة تختلف عن سابقاتها ،فمن أجل تعريف متغير **X** من نوع **String** يجب:

١. تضمين المكتبة **string**

٢. إلغاء **.h** من اسم المكتبة

٣. إضافة أمر تحديث المكتبات **using namespace std;** وهو يشمل أوامر مكتبة **C++** القياسية والمكتبات الجديدة

ملاحظة هامة:

إذا استخدمت في برنامج ما مكتبة **String** واستخدمت في نفس البرنامج المكتبات القديمة مثل **stdio.h** و**Math.h** لا يتم إزالة **.h** من اسم المكتبة .

```
#include<iostream>
#include<math.h>
#include<string>
using namespace std;

    main()
    {
    string x;
    x="ali";
    cout<<x<<endl;
    cout<<sqrt(5);
    return (0);
    }
```

برامج تطبيقية على ماسبق:

برنامج يستقبل رقمين من المستخدم ويجمعهما ويعرض ناتج الجمع

er.cpp

```
// Program: برنامج لجمع عددين
#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;
     }
```

C:\Windows\system32\cmd.exe - er

```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\sd>cd\

C:\>cd c++

C:\C++>cd debug

C:\C++\Debug>er
Enter first integer
7
Enter second integer
8
sum=15
```

برنامج يطبع الرسالة التالية

```
---------- welcome to  c++ ----------
**********
#include<iostream.h>
main()
{
cout<<"          **********\n";
cout<<"----------  welcome to  c++  ----------\n";
cout<<"          **********\n";
}
```

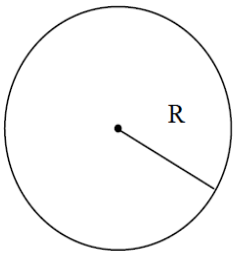
برنامج للإعلان عن عدد صحيح وعدد حقيقي

```
#include<iostream.h>
main()
{
int a=100;
float b=5.36;
cout<<a<<b;
return(0);
}
```

برنامج حساب مساحة الدائرة بمعلومية نصف القطر

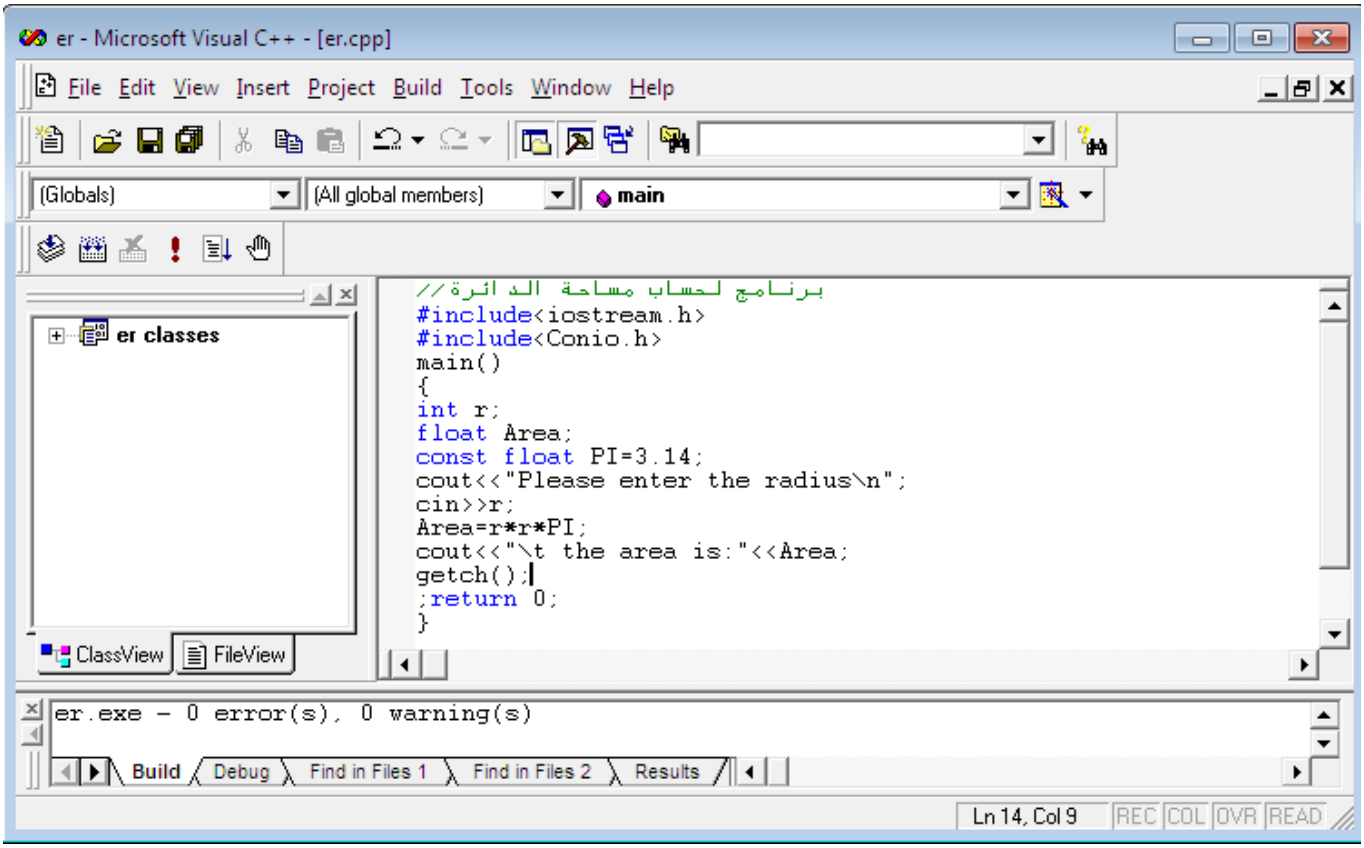
مساحة الدائرة :ط نق^{2}

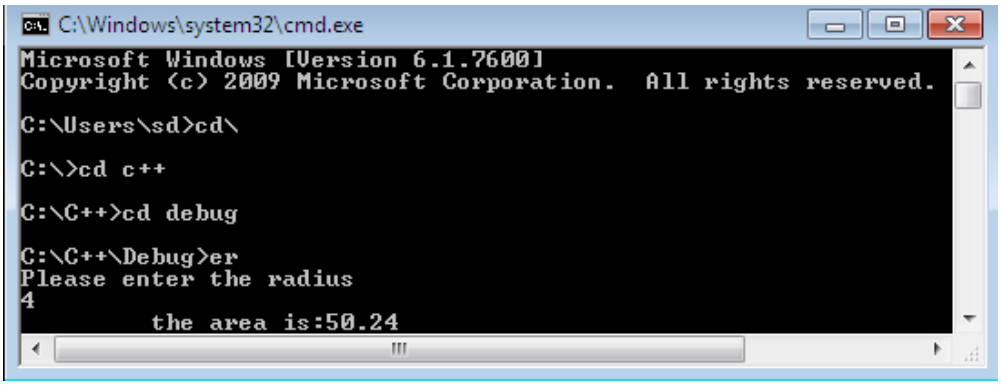
أكتب برنامج لحساب مساحة الدائرة إذا علمت أن:



$$Circle = \pi * R^2$$

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



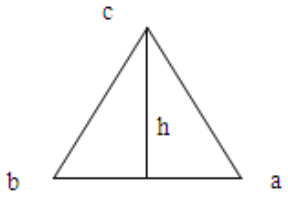


برنامج لحساب محيط الدائرة بمعلومية القطر

```
#include<iostream.h>
main()
{
const float pi=3.14;
int d;
cout<<"circumference of a circle \n"; //=pi*diameter.
cout<<"Enter a diameter \n";
cin>>d;
cout<<pi*d<<endl;
}
```

برنامج حساب مساحة المثلث

مساحة المثلث: $\frac{1}{2}$ (القاعدة×الإرتفاع)



er.cpp

```
//برنامج لحساب مساحة المثلث
#include<iostream.h>
main()
{
int base,height;
cout<<"enter the base of triangle\n";
cin>>base;
cout<<"enter the height of triangle\n";
cin>>height;
cout<<"the area is: "<<0.5*base*height;
return 0;
}
```

C:\Windows\system32\cm...

```
C:\C++\Debug>er
enter the base of triangle
6
enter the height of triangle
2
the area is: 6
C:\C++\Debug>^C
```

برنامج حساب متوسط خمسة أعداد يقوم المستخدم بإدخالها

```
#include<iostream.h>
main()
{
int a,b,c,d,z;
cout<<"Eter five numbers \n";
cin>>a>>b>>c>>d>>z;
cout<<(a+b+c+d+z)/5<<endl;
}
```

برنامج حل معادلة حسابية

$$Z = pr \% q + \frac{w}{x} - y z$$

```
#include<iostream.h>
main()
{
float Z;
int pr,q,w,x,y,z;
cout<<"Enter vaules for pr,q,w,x,y,z\n";
cin>>pr>>q>>w>>x>>y>>z;
Z=((pr%q)+(w/x)-(y*z));
cout<<Z;
}
```

برنامج يطبع القيم التالية

```
#include <iostream.h>
void main()
{
      int i,j;
      for (i=1;i<=5;i++)
      {
            for (j=1;j<=i;j++)
                  cout<<i;
            cout<<endl;
      }
}
```

النتائج :

```
1
22
333
4444
55555
Press any key to continue
```

برنامج به عدة تعليمات:اكتب تعليمه واحده بلغة C++ لتنفيذ مايلي:

١. الإعلان عن المتغيرات c,number,q76354,ThisIsAvariable على أنها من النوع Integer
٢. إظهار رسالة تطلب من المستخدم إدخال عدد صحيح على أن تنتهي الرسالة بإشارة (:) متبوعة بفراغ
٣. قراءة عدد عشري من نوع Float مدخل بواسطة لوحة المفاتيح وتخزن قيمته في المتغير age
٤. اطبع رسالة " This is a c++ Program" على سطر واحد .
٥. اطبع الرسالة السابقة على تضع كل كلمة في سطر مستقل .
٦. اطبع الرسالة السابقة على أن يفصل بين الكلمات مسافة جدولة tab "\t" .

```
#include<iostream.h>
main()
{
int c,This,IsAvariable,q76354,number;
cout<<"Enter value of integer  : ";
float age;
cin>>age;
cout<<"This is a c++ program\n";
cout<<"This \n"<<"is \n" <<"a\n"<< "c++\n" <<" program\n";
cout<<"This\t"<<"is \t"<< "a \t" <<"c++ \t" <<"program \t";
}
```

برنامج به عدة تعليمات :اكتب تعليمه واحده بلغة C++ لتنفيذ مايلي:

١. الإعلان عن المتغيرات X,Y,Z,result على أنها من النوع Integer
٢. إظهار سالة للمستخدم بأن يدخل ٣ أعداد صحيحة
٣. يخزن قيم الأعداد الصحيحة في المتغيرات x,y,z
٤. قم بحساب ضرب المتغيرات ببعضها وإظهار النتيجة في المتغير result
٥. اطبع الرسالة " The Product is "متبوعة بنتيجة المتغير result

```
#include<iostream.h>
main()
{
int x,y,z,result;
cout<<"Enter three integer ";
cin>>x>>y>>z;
result =x * y * z;
cout<<"the product is" << result << '\n';
return 0;
}
```

ماالخطأ في الإعلان عن المتغيرات التالية:

Int first = 22, 1ast=99, new = 44, old =66;

الخطأ هو Int والتعريف الصحيح هو int

المتغيرات لا تبدأ بأرقام في لغة C++

ما الخطأ في البرنامج التالي مع تصحيحه وكتابته بالشكل الصحيح

```
#include<iostream.h>
main()
{
N=2;
cout<<"N="<<N;
return(0);
}
```

الخطأ هوعدم تعريف المتغير N أنه من نوع Integer

```
#include<iostream.h>
main()
{
int N=2;
cout<<"N="<<N;
return(0);
}
```

برنامج يتم فيه إدخال قيمة الزاويه ويقوم البرنامج بإيجاد جيب تمام الزاوية

```
#include<iostream.h>
#include<math.h>
main()
{
float A;
cout<<"Enter A \n";
cin>>A;
cout<<cos(A)<<"\n";
return(0);
}
```

اكتب برنامج يقرأ عددين من نوع Float ثم يختبر العددين هل أحدهما أو كلاهما ذو قيمة سالبة ثم إيجاد جذر مجموع هذين العددين وطباعة العبارة "Error Netgative Numbers" والخروج من البرنامج إذا كان غير ذلك.

```
#include<iostream.h>
#include<math.h>
main()
{
float a,b;
cout<<"Enter two number to test it if negative or positive \n";
cin>>a>>b;
if(a>=0  &&  b>=0)
cout<<sqrt(a+b);
else
cout<<"Error Negative Numbers \n";
}
```

برنامج يقوم بطباعة مربع ومكعب وجذر العدد المدخل

```
#include<iostream.h>
#include<math.h>
main()
{
int a;
cout<<"Enter any number \n";
cin>>a;
cout<<pow(a,2)<<"\n";
cout<<pow(a,3)<<"\n";
cout<<sqrt(a);
  }
```

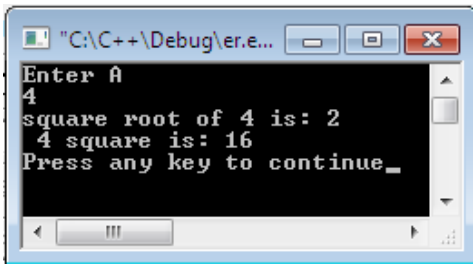
برنامج لحل معادلة حسابية

$$z=\sqrt{b\wedge 2-4*a*c}$$

```
//برنامج لحل معادلة
#include <iostream.h>
#include <math.h>
main()
{
int a,b,c;
double z;
cout<< "Enter a,b,c \n";
cin>> a;
cin>> b;
cin>> c;
z=sqrt(b*b-4*a*c);
cout<<"The  Root ="<<z;
return 0;
}
```

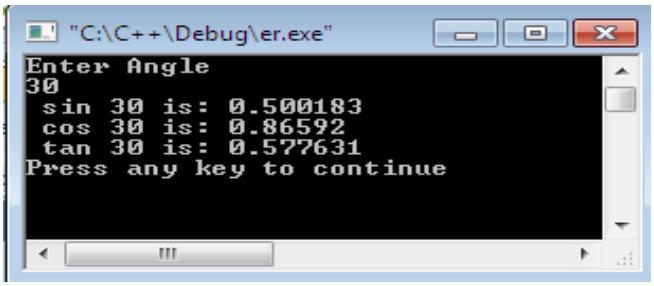
برنامج لإدخال رقم وإيجاد مربعه والجذر التربيعي له

```
//برنامج لادخال رقم وإيجاد مربعه والجذر التربيعى له
#include <iostream.h>
# include <math.h>
int main()
{
int a;
cout<< "Enter A \n";
cin>> a;
cout<<"square root of " << a << " is: " << sqrt(a) <<" \n";
cout<< " "<< a << " square is: " << a * a <<" \n";
}
```



برنامج لإدخال قيمة زاوية وإيجاد قيم الدوال المثلثية الثلاث (Sin – Cos – Tan) لها

```
//برنامج لادخال قيمة زاوية وإيجاد
//قيم الدوال المثلثية الثلاث (Sin - Cos - Tan)لها
#include <iostream.h>
# include <math.h>
int main( )
{
int a;
cout<< "Enter Angle \n";
cin>> a;
const float pi =3.142857;
float s = sin (a* pi /180), c = cos (a* pi /180),t = tan (a* pi /180);
cout<< " sin " << a << " is: " << s <<" \n";
cout<< " cos " << a << " is: " << c <<" \n";
cout<< " tan " << a << " is: " << t <<" \n";
}
```



جمل التحكم الشرطية (القرارات) Condition Statements

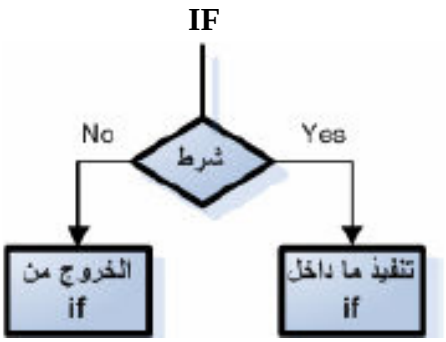
أولاً :جملة IF

يتألف التعبير IF من الكلمة الأساسية IF، يليها جسم القرار ويتكون إما من عبارة واحدة، أو من عدة عبارات تحيطها أقواس { }،ويتم تنفيذ ما بين الأقواس في حالة تحقق الشرط أما في حالة عدم تحقق الشرط فلا يتم تنفيذ شيء.

If (condition)

Statement 1;

يتم تنفيذ جملة واحدة في حالة تحقق الشرط
"لا يوجد أقواس لأنها جملة واحدة"



شكل يوضح تركيب جملة IF

If (condition)
{
Statement 1;
Statement 1;
Statement 1;
.
}

يتم تنفيذ عدة جمل في حالة تحقق الشرط
" يوجد أقواس لأنها عدة جمل"

ملاحظات:

- لا يشترط كتابة الأقواس إذا كانت جملة واحدة .مثال:

```
If (X == 100)
Cout<< " x is 100" ;
```

- هذه الجمل الموجودة بين الأقواس لا يتم تنفيذها إلا في حالة تحقق الشرط.
- التعبير IF يعني "إذا حصل هذا الشيء سأفعل كذا وكذا "

برنامج يقوم بطباعة رسالة حسب العمر

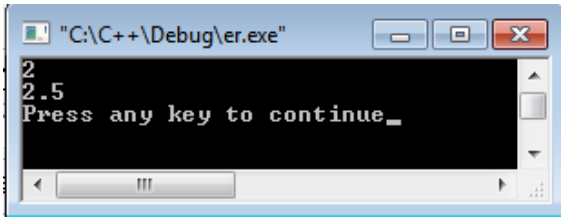
```
#include<iostream.h>
main()
{
     int a;
     cout<<"Enter your age\n";
     cin>>a;
     if(a>0 && a<18)
          cout<<"you are child\n";
     if(a>=18 && a<=65)
          cout<<"your are adult\n";
     if(a>65)
          cout<<"your are senescent\n";
}
```

برنامج يستقبل عددين ويطبع رسالة إذا كانت الأرقام متساوية

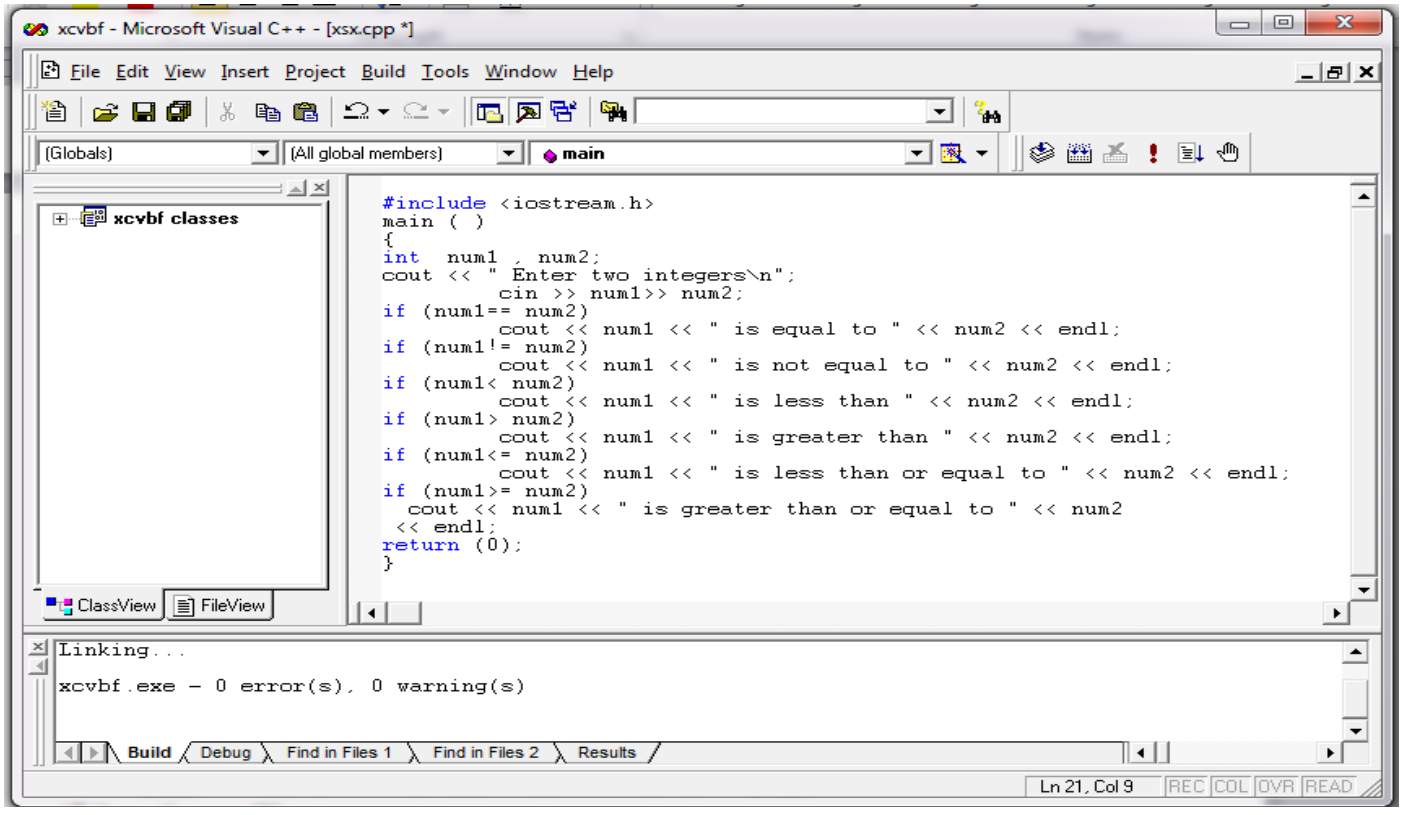
```
#include<iostream.h>
main( )
{
      int a,b;
      cout<<"Enter two numbers \n";
                              cin>>a>>b;
      if(a==b)
            cout<<a<<"is equal to "<<b<<endl;
      else
            cout<<a<<"is not equal to "<<b<<endl;
}
```

برنامج يوضح أنواع المتغيرات والخرج من كل منها

```
#include <iostream.h>
void main ( )
{
int i ;
float z;
i = 5/2;
z = 5. / 2 ;
cout<<i <<"\n"<< z<<endl;
}
```

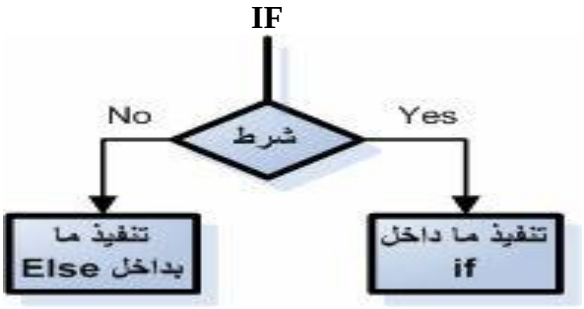


برنامج مقارنة بين عددين باستخدام IF



ثانياً:جملة IF-Else

يتألف التعبيرIF-Else من الكلمةالأساسية IF، يليها عبارة أو مجموعة عبارات يتم تنفيذها في حالة ما إذا كان الشرط صحيح ثم الكلمة Else يليها عبارة أو مجموعة عبارات يتم تنفيذها في حالة عدم تحقق الشرط.



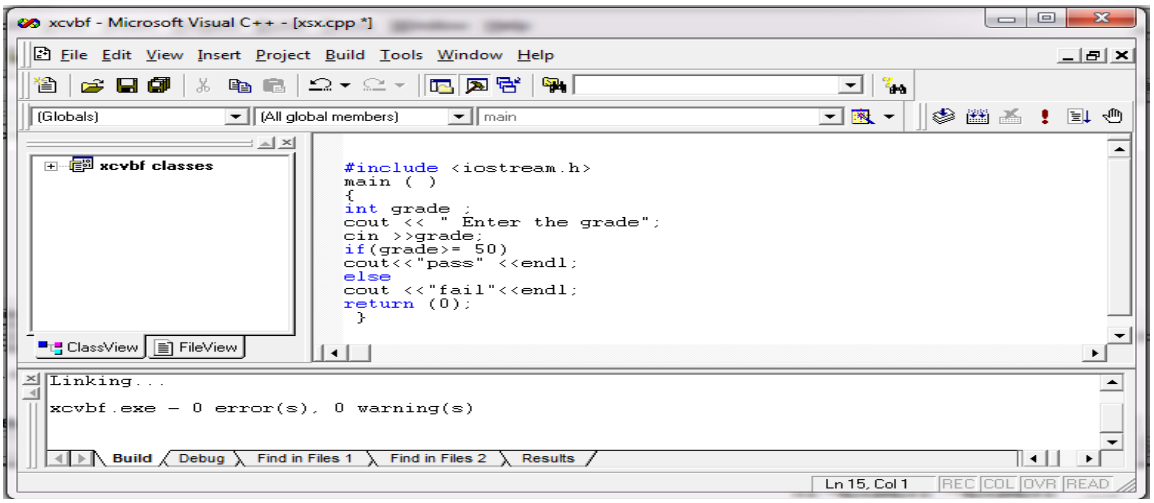
شكل يوضح تركيب جملة IF-Else

```
if (condition)
{
Statement 1;
Statement 1;
}
else
{
Statement 1;
Statement 1;
}
```

- جملة IF-Else تعني "إذا حصل هذا الشيء سأفعل كذا وإلا سأفعل كذا "

تطبيقات على IF-ELSE

برنامج يختبر مجموع الطالب وإخراج كلمة Pass إذا كان التقدير أكبر من أويساوي ٥٠



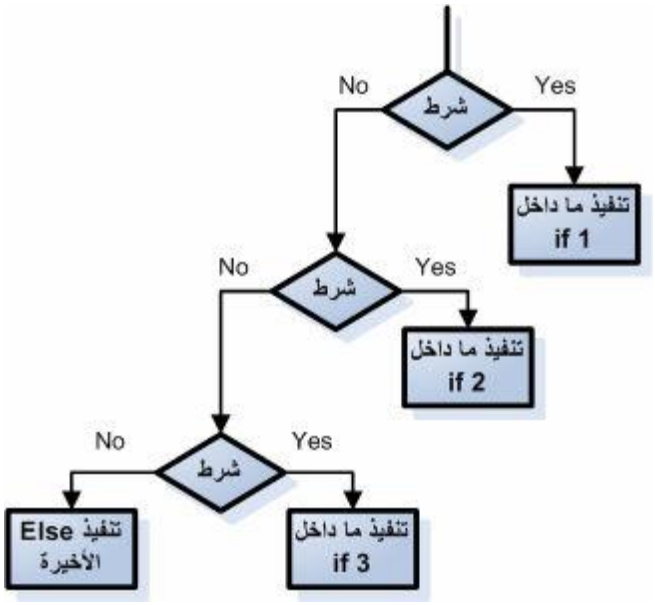
برنامج يختبر هل العدد فردي أم زوجي

```
//برنامج يختبر هل العدد فردي أم زوجي
#include<iostream.h>
main()
{
int a;
cout<<"enter the number \n";
cin>>a;

if(a%2==0) //الرمز % يشير إلى باقي القسمة
cout<<"the number is Even  \n";
else
cout<<"the number is Odd`  \n";
return(0);
}
```

ثالثاً:التعبير..... IF-Else IF – Else IF

ويتألف هذا التعبير من عدة جمل IF-Else متداخله تُستخدم في حالة تعدد الشروط .

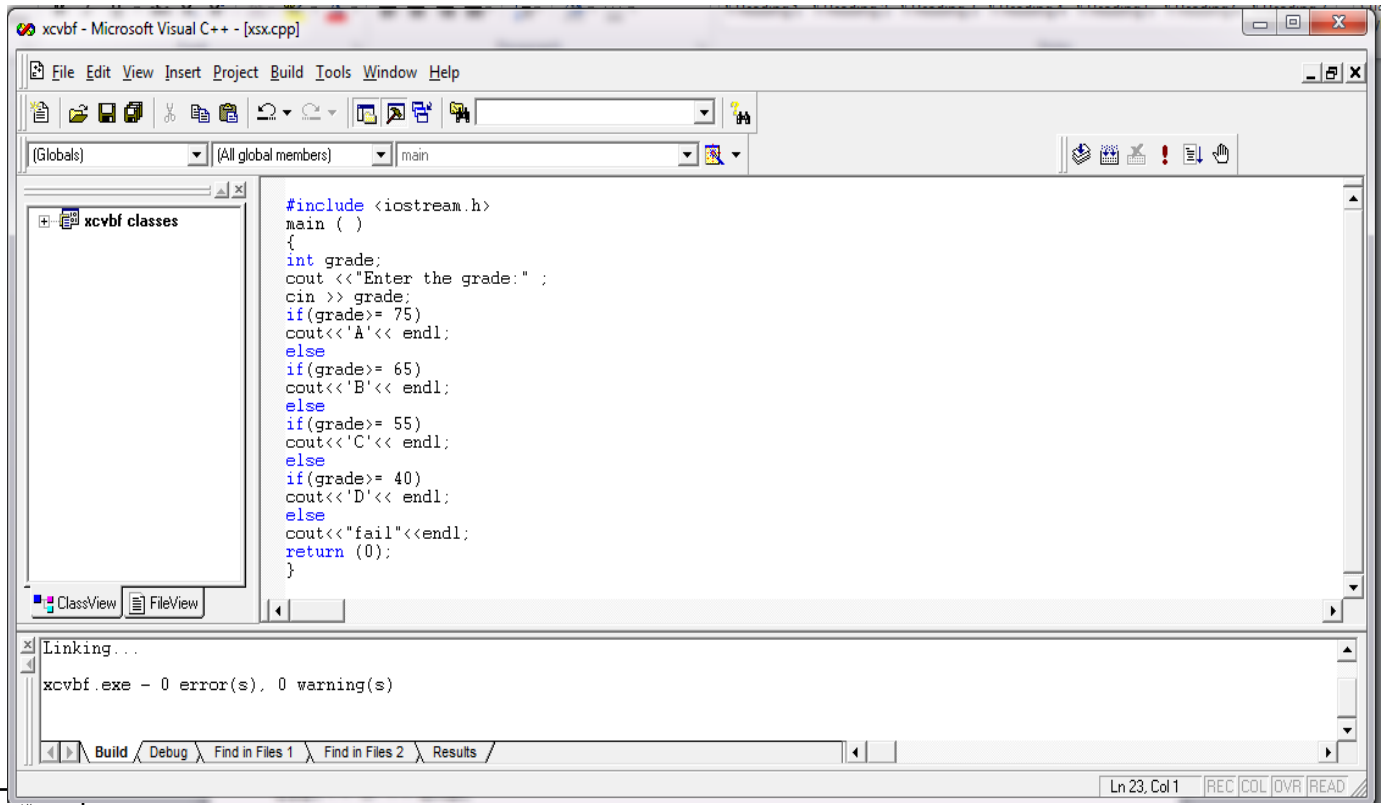
```
if (condition1)
{
Statement 1;
Statement 1;
}
else
if (condition2)
{
Statement 1;
Statement 1;
}
.
```



التعبير ... IF-Else IF –Else IF يعني"إذا تحقق شرط كذا سأفعل كذا وكذا إما إذا تحقق شرط كذا سأفعل كذا وكذا أما إذا تحقق شرط كذا ....." الخ.

تطبيقات على IF-Else IF –Else IF

برنامج يختبر هل العدد فردي أم زوجي

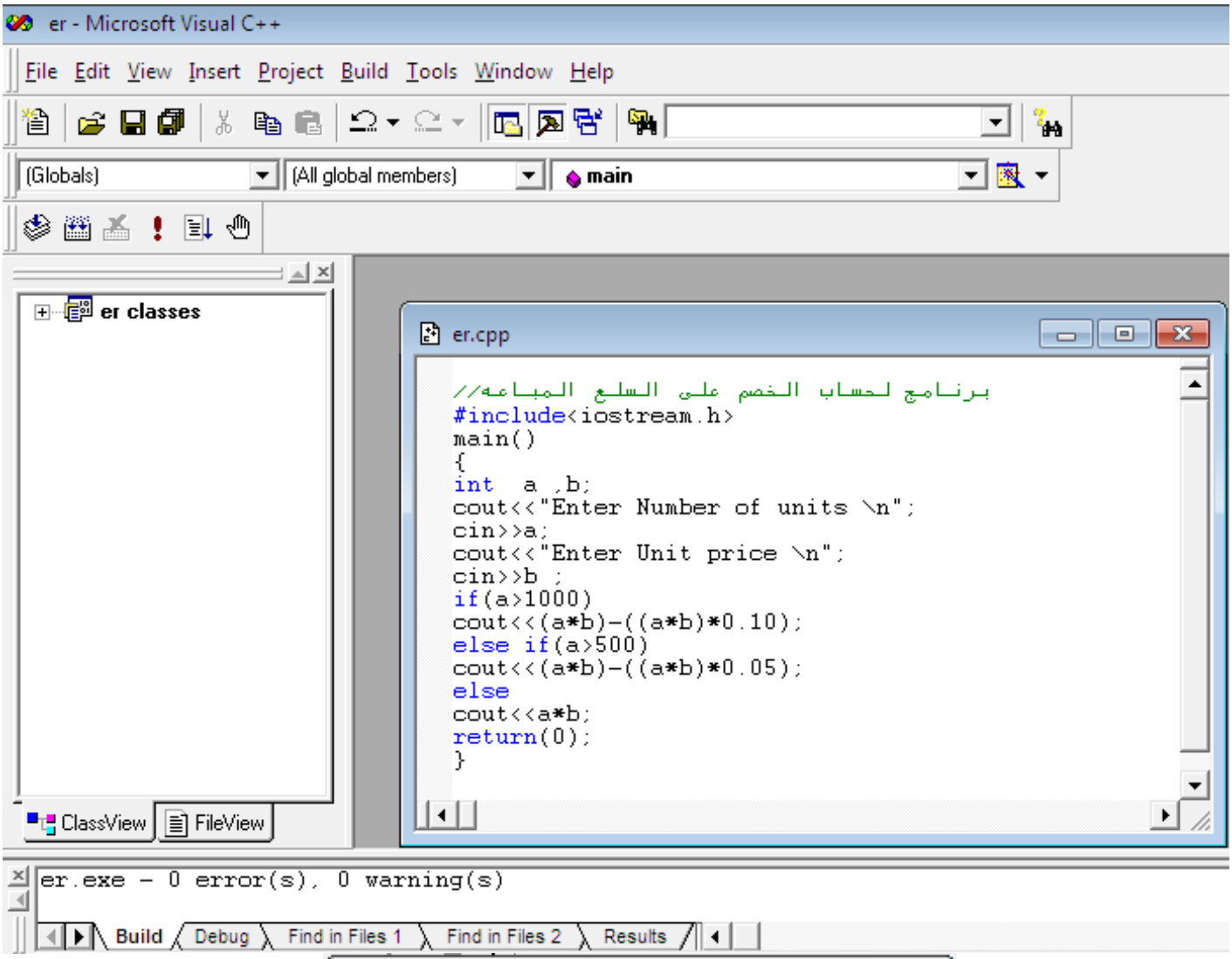


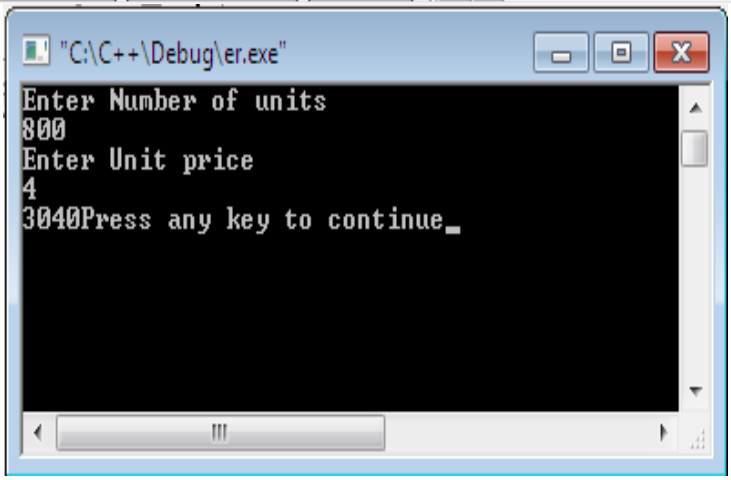
برنامج لحساب الخصم على السلع المباعه

١. خصم ١٠% في حالة بيع اكثر من ١٠٠٠ قطعه

٢. خصم ٥% في حالة بيع اكثر من ٥٠٠ قطعه

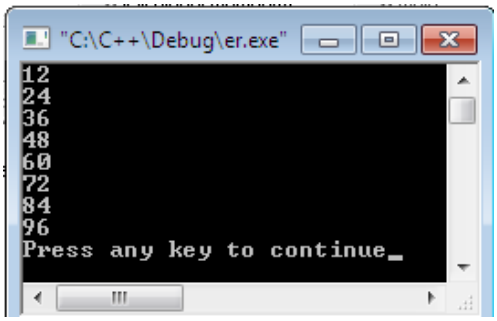
٣. لا يوجد خصم إذا كانت الوحدات المباعه ٥٠٠ فأقل





برنامج يقوم بطباعة الأرقام التى تقبل القسمة على 2،4،6 بدون باقى بين الأعداد من 1 إلى 100

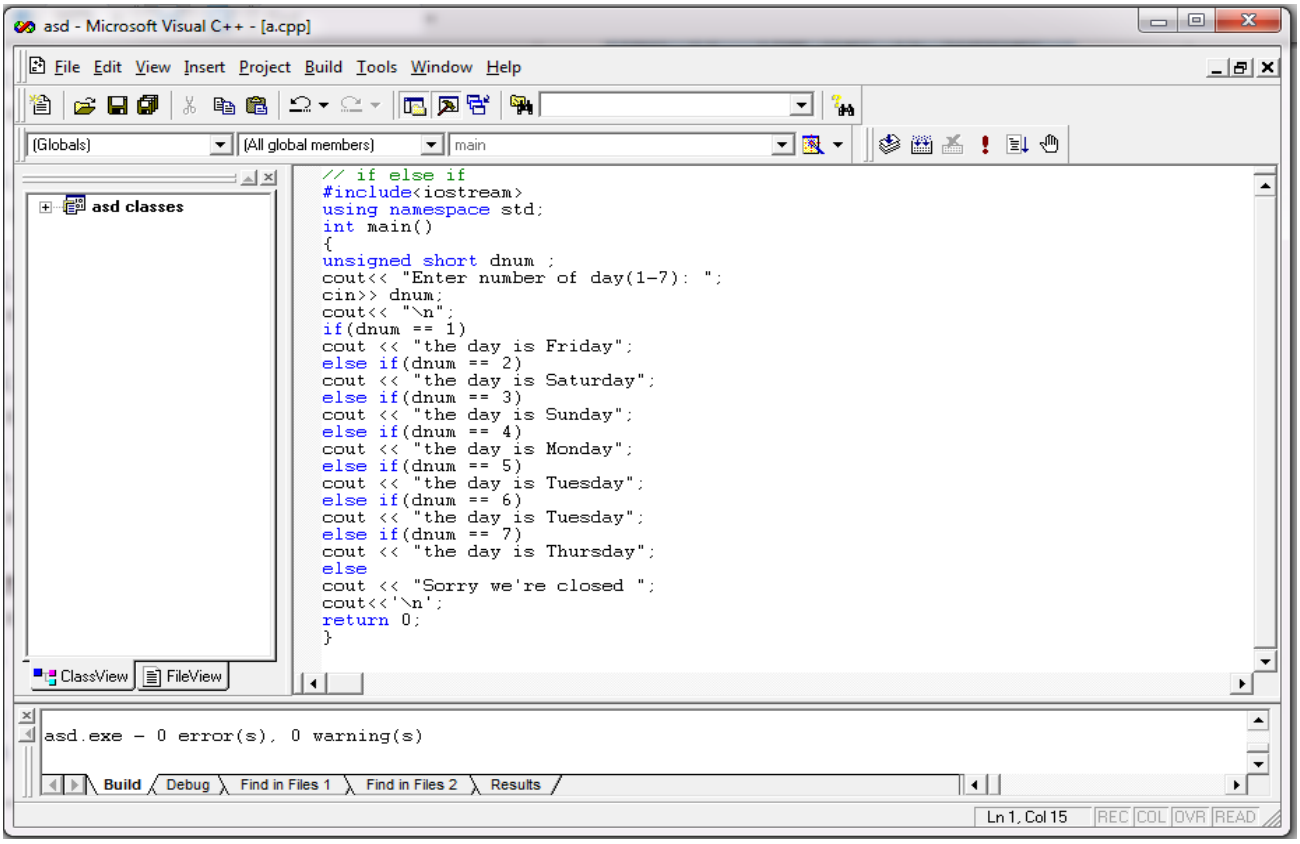
```
#include < iostream.h>
main ()
{
int number;
for (number=1;number<=100;++number)
{
if (number%2)
continue;
else if (number%4)
continue ;
else if (number%6)
continue;
else
cout<<number<<endl;
}
return 0;
}
```



برنامج يطبع رسالة أن قيمة X موجبه "X is Positive"أو سالبه "X is Negative"

```
# include <iostream.h>
main()
{
int x
cout<<"Enter The test number X ";
cin>> x ;
if (x>0)
cout << " x is positive\n";
else if (x<0)
cout<<"x is negative\n";
else if(x= =0)
cout<<" x=0\n";
return 0;
}
```

برنامج يستقبل رقم اليوم ويقوم بطباعة اسم اليوم باللغة الإنجليزية



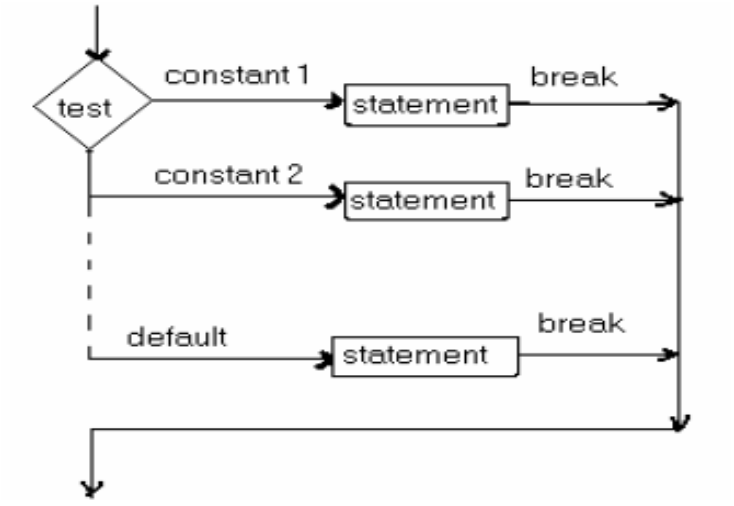

```
// if else if
#include<iostream>
using namespace std;
int main()
{
unsigned short dnum ;
cout<< "Enter number of day(1-7): ";
cin>> dnum;
cout<< "\n";
if(dnum == 1)
cout << "the day is Friday";
else if(dnum == 2)
cout << "the day is Saturday";
else if(dnum == 3)
cout << "the day is Sunday";
else if(dnum == 4)
cout << "the day is Monday";
else if(dnum == 5)
cout << "the day is Tuesday";
else if(dnum == 6)
cout << "the day is Tuesday";
else if(dnum == 7)
cout << "the day is Thursday";
else
cout << "Sorry we're closed ";
cout<<'\n';
return 0;
}
```

رابعاً :التعبير Switch Statement

إذا زاد عدد الاحتمالات فمن الأفضل استخدام عبارة **switch** بدلاً من جملة **IF** المتداخله

الشكل العام لجملة switch

```
Switch (variable)
{
case value1 ;
statement1;
break;
case value2;
statement2;
break;
case value N;
statementn;
break;
default:
statement;
break;
}
```



شرح تركيب جملة switch:

١. تبدأ بكلمة **switch** ثم الـــمُتغير المطلوب اختباره بين قوسين

٢. قوس كبير { يوضح بداية التركيب وبداخله عدة بدائل

٣. كل بديل يبدأ بكلمة **case**(الشرط)يليه جمله أو عدة جمل تُنَفَذ في حالة تحقق البديل وينتهي كل بديل بكلمة **break**

٤. وبعد نهاية بلوكات البدائل تأتي كلمة (**Default**) متبوعة بعبارة أو بعدة عبارات ينفذها الكمبيوتر في حالة عدم تحقق أى من الشروط السابقة(أي أن المستخدم أدخل قيمة خارج دائرة الإختبار) ثم كلمة **break** ثم قوس كبير } يوضح نهاية التركيب.

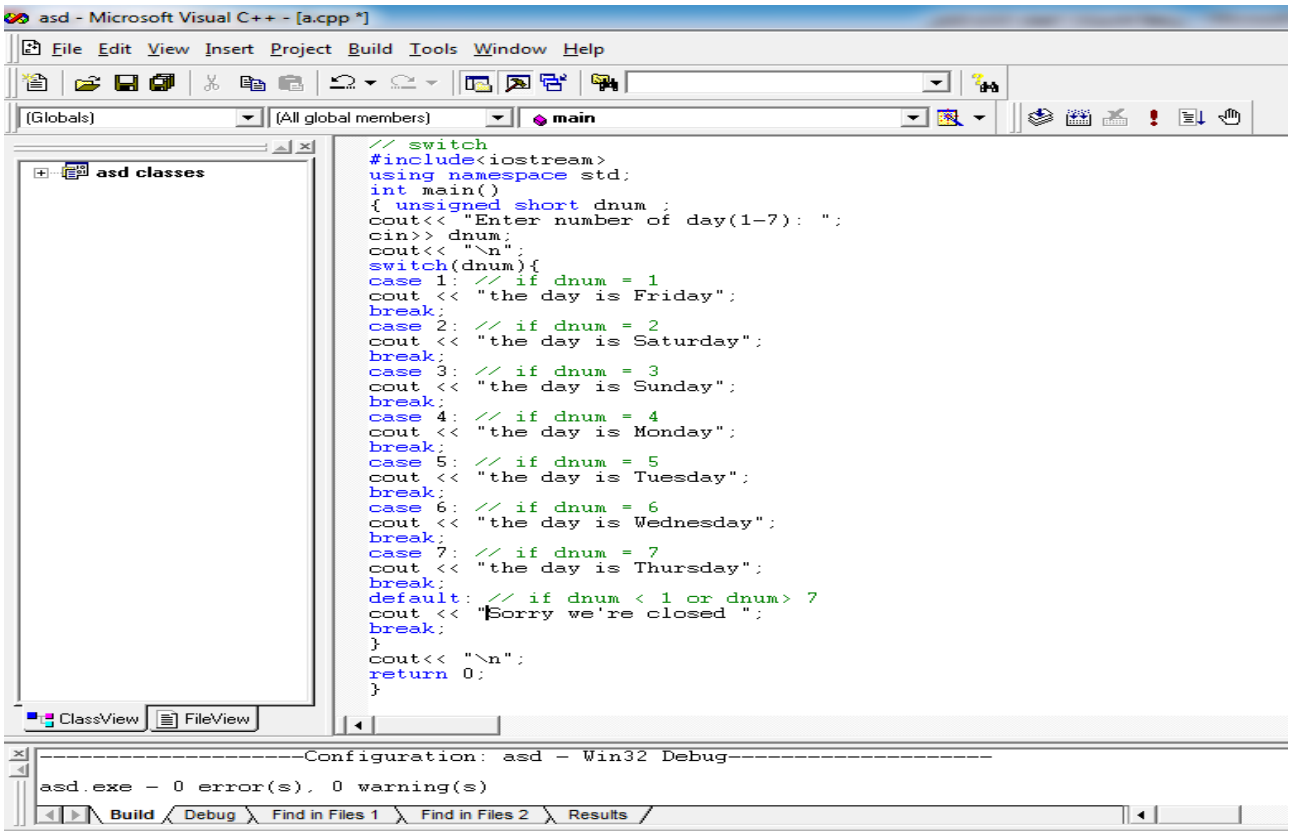
برنامج يستقبل رقم الشهر ويقوم بطباعة اسم الشهر باللغة الإنجليزية

```
#include<iostream.h>
main()
{
int a;
cout<<"enter the month number \n";

cin>>a;

switch (a)
{
case 1:cout<<"January \n";break;
case 2:cout<<"February \n";break;
case 3:cout<<"March  \n";break;
case 4:cout<<"April  \n";break;
case 5:cout<<"May  \n";break;
case 6:cout<<"June  \n";break;
case 7:cout<<"July  \n";break;
case 8:cout<<"August  \n";break;
case 9:cout<<"September  \n";break;
case 10:cout<<"October  \n";break;
case 11:cout<<"November  \n";break;
case 12:cout<<"December  \n";break;
default:
cout<<"error try again \n";break;
}
return(0);
}
```

برنامج لإجراء العمليات الحسابية باستخدام Switch Case(آله حاسبه)

```
#include<iostream.h>
main()
{
int a,b;
cout<<"enter two numbers \n";

cin>>a>>b;
char z;
cout<<"Enter the Operation \n";
cin>>z;
switch (z)
{
case '+':cout<<a+b;break;
case '-':cout<<a-b;break;
case '/':cout<<a/b;break;
case '*':cout<<a*b;break;
default:
cout<<"error try again \n";break;
}
return(0);
}
```

برنامج يستقبل رقم اليوم ثم يقوم بطباعة اسم الشهر باللغة الإنجليزية باستخدام التعبير switch



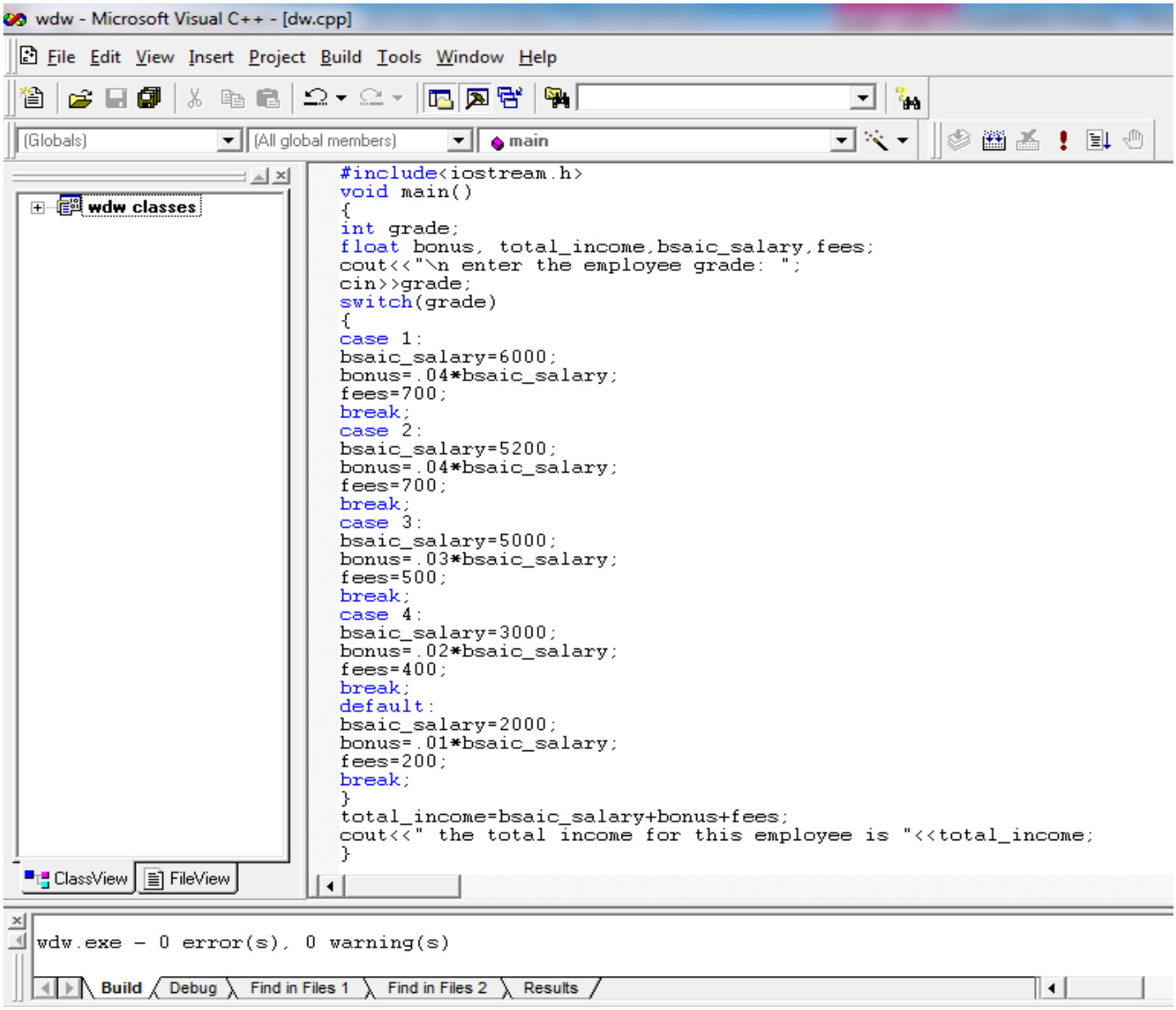
برنامج يحول من الأرقام العربيه إلى الأرقام الموافقه لها من الرومانية باستخدام جملة Switch

```
#include<iostream.h>
main()
{
int x;
cout<<"enter the decimal number:";
cin>> x;
while(x<=3)
{switch(x)
{
case 1:
{cout<<"I"<<endl;
cout<<"enter the decimal number:";}
break;
case 2:
{cout<<"II"<<endl;
cout<<"enter the decimal number:";}
break;
case 3:
{cout<<"III"<<endl;
cout<<"enter the decimal number:";}
break;
default:
cout<<"NOOOO";
break;
}
cin>>x;
}
return(0);
}
```

اكتب برنامج بلغة C++ يقوم بحساب الدخل الكلي للموظف Total_income،إذا علمت درجته الوظيفية حيث يحسب الدخل الكلي بالمعادلة:

Total_Income =Basic_salary+Bonus+Fess حيث **Basic_salary** هو الراتب الاساسي،**Bonus** يمثل العلاوه،**Fess** تمثل البدلات وقيم هذه المتغيرات تعتمد على الجدول التالي:

grade	bsaic_salary	bonus	fees
1	6000	=.04*bsaic_salary	700
2	5200	=.04*bsaic_salary	700
3	5000	=.03*bsaic_salary	500
4	3000	=.02*bsaic_salary	400
Other grades	2000	=.01*bsaic_salary	200



حلقات التكرار (loops)

توفر C++ عدداً من أساليب التكرار (حلقات) التي تُستخدم لتكرار أجزاء من البرنامج طالما أن تعبير الإختبار صحيح وإلا يتوقف عن التكرار .

هناك ثلاثة أنواع من الحلقات في C++:-

أولاً:الحلقة FOR

- في الحلقة **For** يكون عدد مرات تنفيذ الحلقة مذكوراً في بدايتها،والأوامر التي تحتاج لتكرار توضع في جسم الحلقة.
- تحتاج الحلقة **For** إلى عداد(رقم تبدأ منه الدوران ورقم تنتهي إليه)لكي تُنفذ الدورات ومقدار الزيادة .

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
```

$$x{+}{+} \iff x=x+1$$
$$x{-}{-} \iff x=x-1$$

الفرق بين ++i و i++ :

مثال ١:

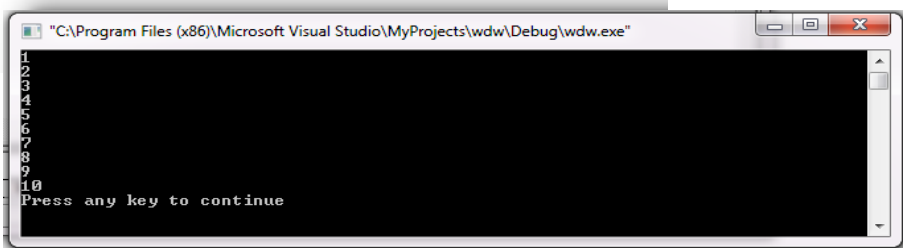
```
cout << i ++;
cout << ++ i;
```

طباعة i ثم زيادته بمقدار واحد.
زيادة i بمقدار واحد ثم طباعته.

تطبيقات على جملة FOR

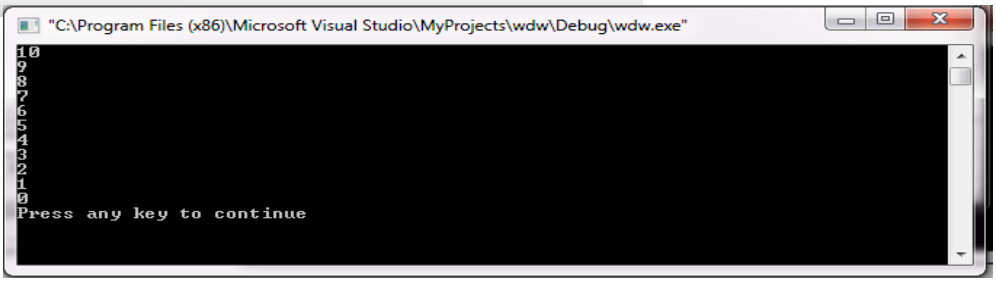
المثال التالي يقوم بطباعة قيم المتغير **counter** تزايدياً من ١ إلى ١٠ .

```
#include<iostream.h>
main( )
{
for ( int counter=1; counter<= 10; counter++)
cout << counter <<endl;
return (0);
}
```

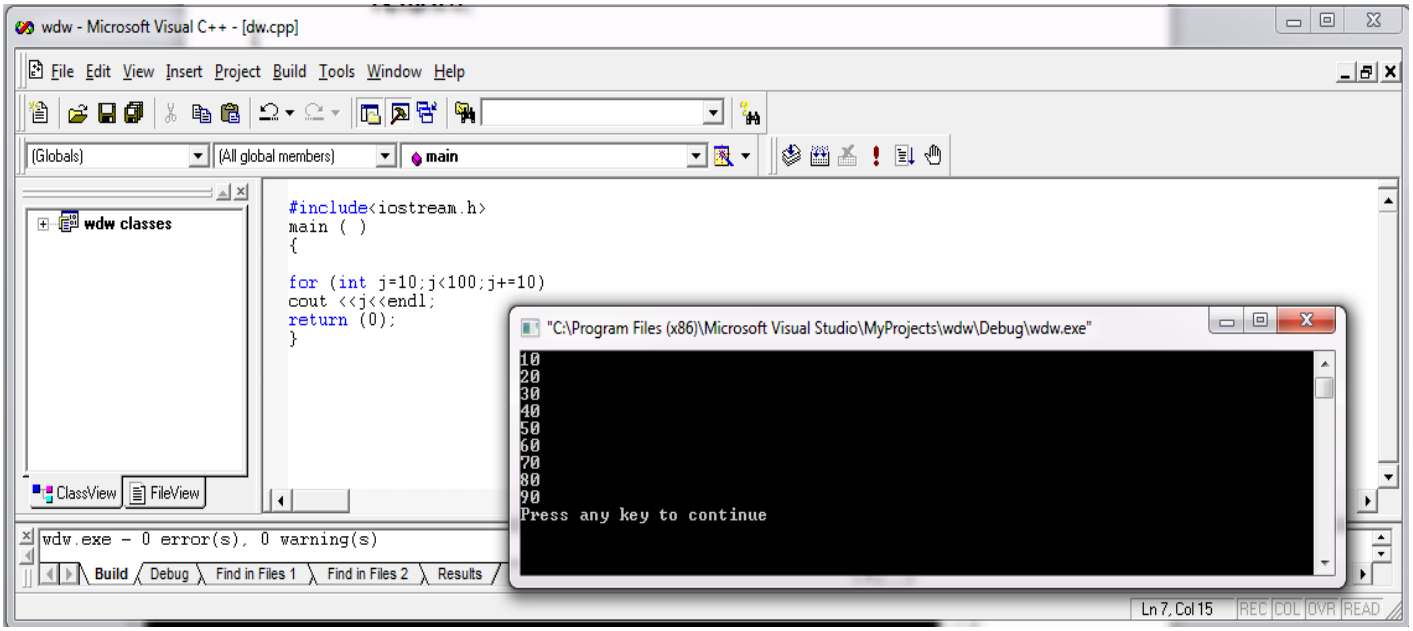


المثال التالي يقوم بطباعة قيم المتغير **counter** تناقصياً من ١٠ إلى ١

```
#include<iostream.h>
main( )
{
for ( int counter=10; counter>= 0; counter--)
cout << counter <<endl;
return (0);
}
```



المثال التالي يقوم بطباعة قيم المتغير **j** بزيادة ١٠ في كل مرة



برنامج يطبع الأعداد الفردية من ١ إلى ١٠٠ باستخدام جملة **For**

```
#include < iostream.h>
main ( )
{
int a;
for (a=1;a<=100;a=a+2)
cout<<a<<endl;
return 0;
}
```

برنامج لطباعة الأعداد من صفر إلى ٩٩ مرتبه تصاعديا باستخدام جملة For

```
//برنامج لطباعة 100 عدد من صفر إلى 99
#include <iostream.h>
int main()
 {
 int x;
 for(x=0;x<100;x++)
 {
  cout<<x<<endl;
 }
  return 0;
 }
```

اكتب برنامج بلغة c++ لإيجاد مضروب الأعداد من ١ إلى ١٠ باستخدام For

```
#include <iostream.h>
void main()
{
	int n,i,fact;
	for (n=1;n<=10;n++)
	{
		fact=1;
		for (i=1;i<=n;i++)
		fact=fact*i;
		cout<<"fact "<<n<< "="<<fact<<endl;
	}

}
```

النتائج o

```
fact 1=1
fact 2=2
fact 3=6
fact 4=24
fact 5=120
fact 6=720
fact 7=5040
fact 8=40320
fact 9=362880
fact 10=3628800
Press any key to continue
```

اكتب برنامج بلغة c++ ليحدد أكبر قيمة من بين ثلاث قيم ويكرر هذه العملية لـ n من المجموعات

```
#include <iostream.h>
void main()
{
		int x,y,z,k,i,max;
		cin>>k;
		for (i=1;i<=k;i++)
		{
			cin>>x>>y>>z;
			max=x;
			if (y>max)
				max=y;
			if (z>max)
				max=z;
			cout<<"maximum of group "<<i<<" = "<<max<<endl;
		}
}
```

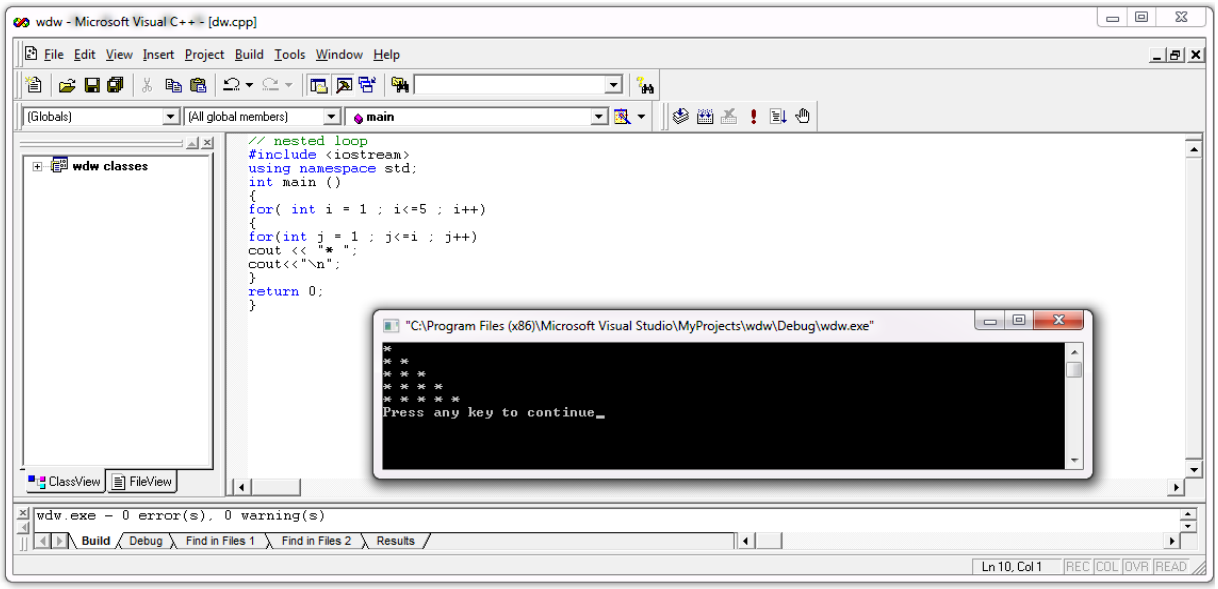
الحلقات المتداخلة Nested Loops :

تأخذ الحلقات For المتداخلة الشكل العام التالي :-

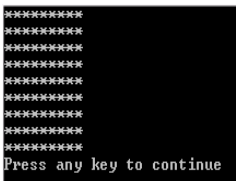
```
for (.....)
for (.....)
for (.....)
Statements;
```

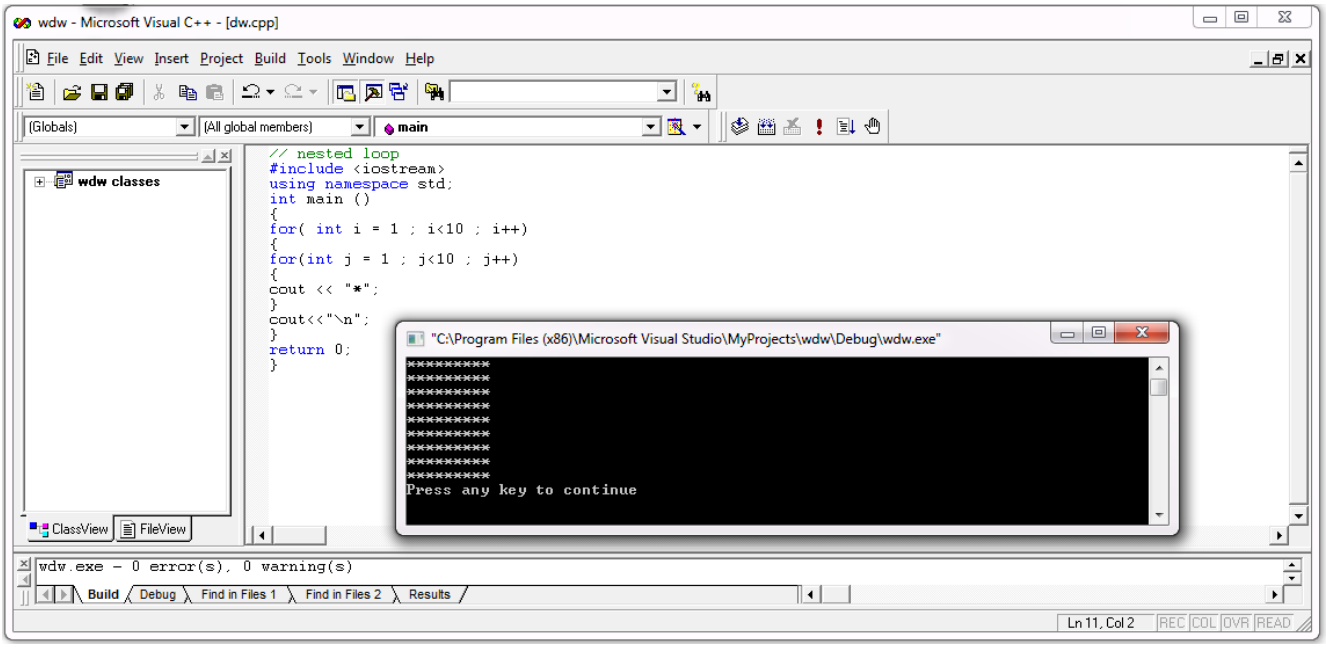
كيف تحصل على الشكل التالي باستخدام جملة FOR المتداخلة

```
*
* *
* * *
* * * *
* * * * *
```



كيف تحصل على الشكل التالي باستخدام جملة For المتداخلة





ثانياً:الحلقة While

تتيح الحلقة **while** تكرار فعل جزء من البرنامج إلى أن يتغير شرط ما .

فمثلاً:-

```
While (n<100)
n=n*2
```

ستستمر هذه الحلقة في مضاعفة المتغير **n** إلى أن تصبح قيمة **n** أقل من أو تساوي **100** عندها تتوقف.

- تتكون الحلقة من الكلمة الأساسية **while** يليها تعبير اختبار بين أقواس ويكون جسم الحلقة محصوراً بين أقواس حاصرة **{ }** إلا إذا كان يتألف من عبارة واحدة.
- مما يجدر التنويه إليه هنا أنه يتم فحص تعبير الاختبار قبل تنفيذ جسم الحلقة، وعليه لن يتم تنفيذ جسم الحلقة أبداً إذا كان الشرط خطأ عند دخول الحلقة وعليه المتغير **n** في المثال السابق يجب تمهيده عند قيمة أقل من **100** .
- تستمر الحلقة بلا توقف طالما الشرط متحقق

الشكل العام لحلقة **while**

```
While (condition)
{
Statements (جسم الحلقة)
}
```

مثال :

الكود:

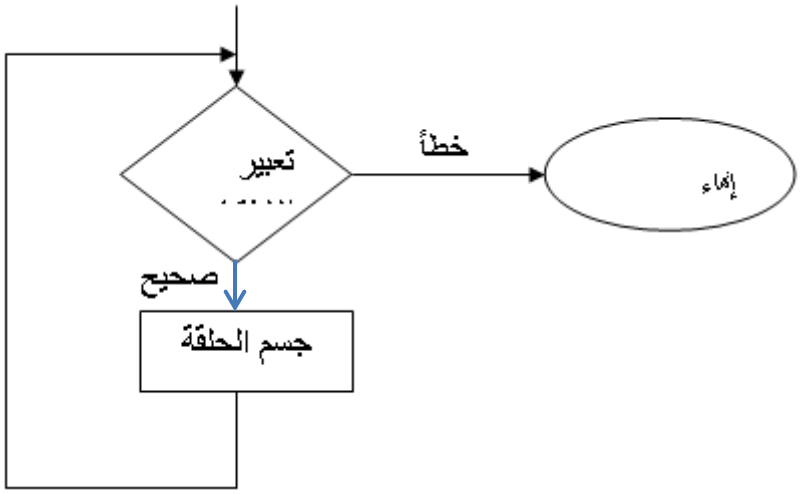
```
1.  int w = 3;
2.  while ( w <=3 )
3.  {
4.     cout << "value is : " << w << endl;
5.     w++;
6.  }
```

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.

شكل يوضح تركيب حلقة while



شكل يوضح تركيب الحلقة while

برنامج يطبع الأعداد من ١ إلى ١٠ بجملة While

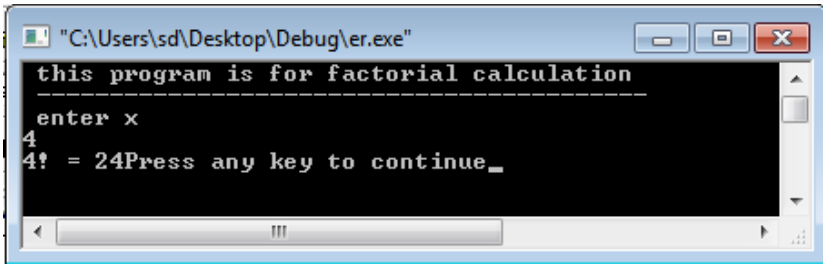
```
#include<iostream>        //for cin cout
#include<conio.h>         //for getch()
using namespace std;
//*****************************************************
void main()
{
     int i=0;
     while(i<=10)
     {
          cout<<"the number = "<<i<< " "<<endl;
          i++;           //  الزيادة بمقدار واحد
     }
     getch();            //  هذا الامر حتى لاتغلق الشاشة بسرعة
}
```

برنامج يطبع الأعداد من ١ إلى 99 بجملة While

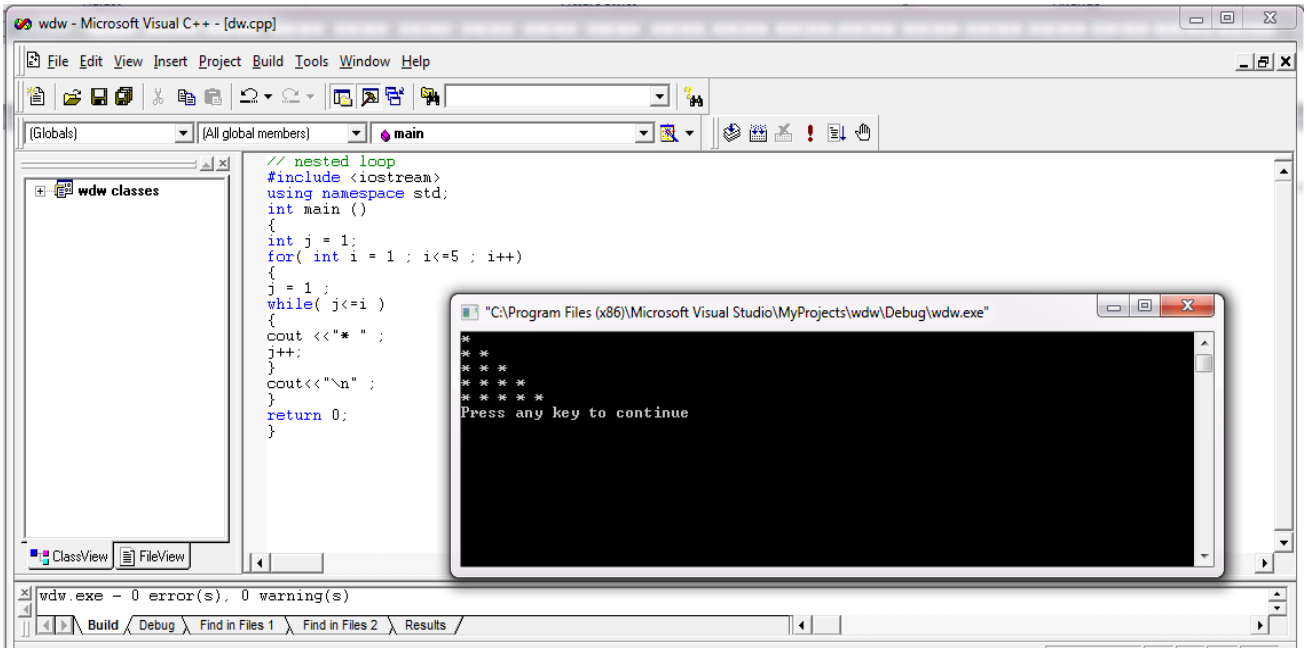
```
#include <iostream.h>
int main()
{
int x=0;
while(x<100)
{
cout<<x<<endl;
x++;
}
return 0;
}
```

برنامج لحساب مضروب العدد باستخدام جملة while

```
#include<iostream.h>
void main()
{
int x,y,fact=1;
cout<<" this program is for factorial calculation";
cout<<"\n ----------------------------------------";
cout<<"\n enter x"<<endl ;
cin>>x;
y=1;
cout<<x<<"! = ";
while(y<=x)
{
fact=fact*y;
y++;
}
cout<<fact;
}
```



حلقة For مع While

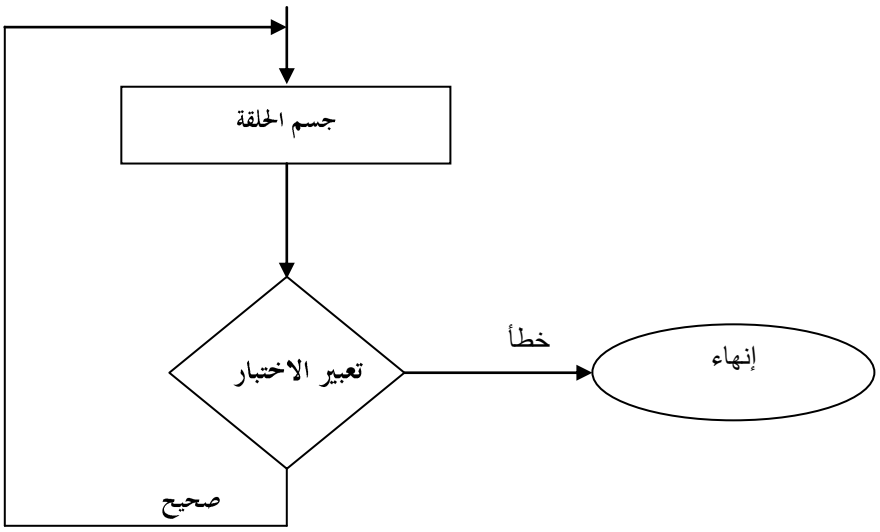


ثالثاً:الحلقة Do-while

تعمل الحلقة (do…while) كالحلقة while،إلا أنها تفحص تعبير الاختبار بعد تنفيذ جسم الحلقة. وتستخدم أيضاً عندما نريد القيام بجزء من البرنامج مرة واحدة على الأقل.

الشكل يبين كيفية عمل الحلقة do-while.

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



شكل يوضح تركيب الحلقة Do - while

تبدأ الحلقة do بالكلمة الأساسية do يليها جسم الحلقة بين أقواس حاصرة { } ثم الكلمة الأساسية while ثم تعبير اختبار بين أقواس ثم فاصلة منقوطة.

مثال :

النتيجة:

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

الكود:

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

Do while تقوم بتنفيذ الكود مرة واحده حتى لو كان الشرط خاطئاً

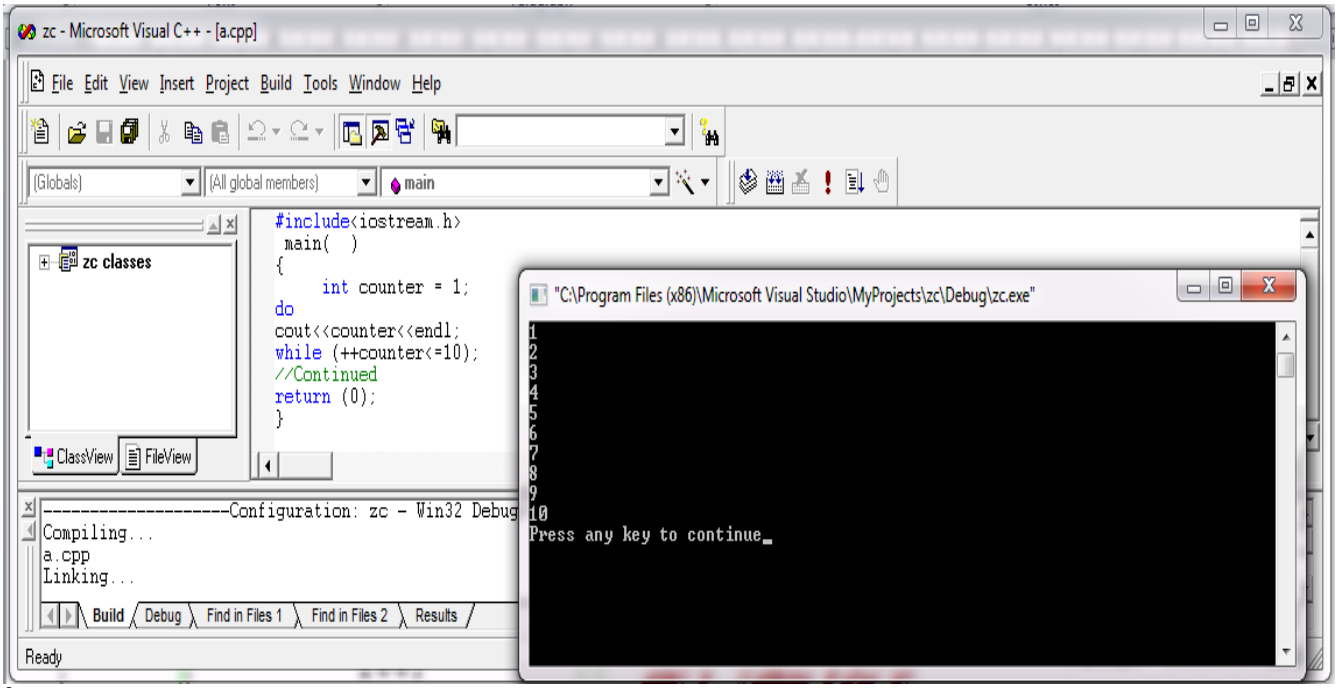
النتيجة:

```
value is: 0
```

الكود:

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

برنامج يطبع الأعداد من ١ إلى ١٠ باستخدام جملة Do- While

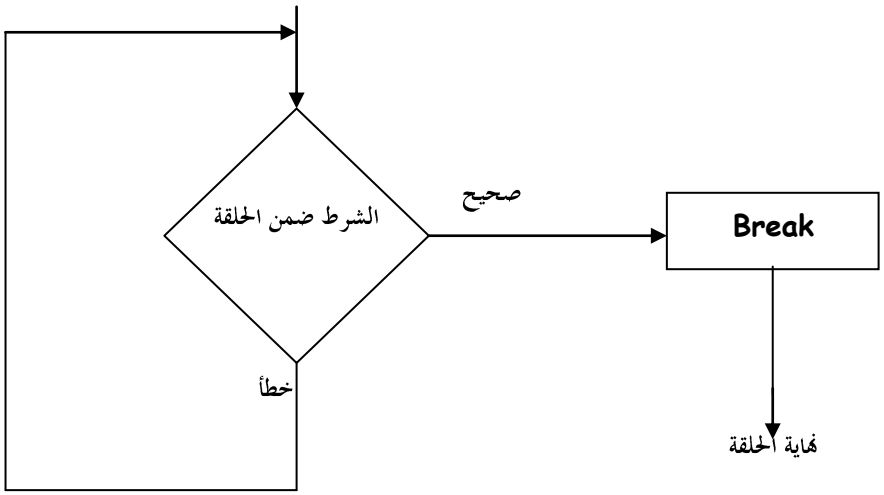


طرق التحكم في حلقات التكرار

١. التعليمة Break

تتيح لنا التعليمة **break** الخروج من الحلقة في أي وقت في حالة تحقق شرط معين وعند تنفيذها يتم القفز إلى سلسلة الجمل التالية للبرنامج.

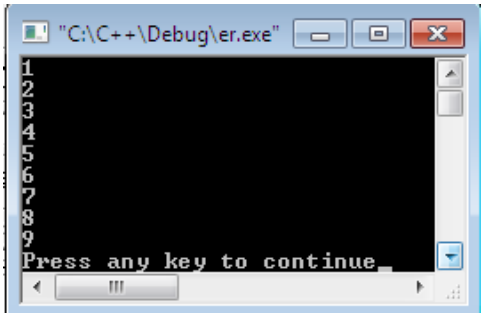
الشكل التالي يبين طريقة عمل العبارة **break**:-



طريقة عمل العبارة **Break**

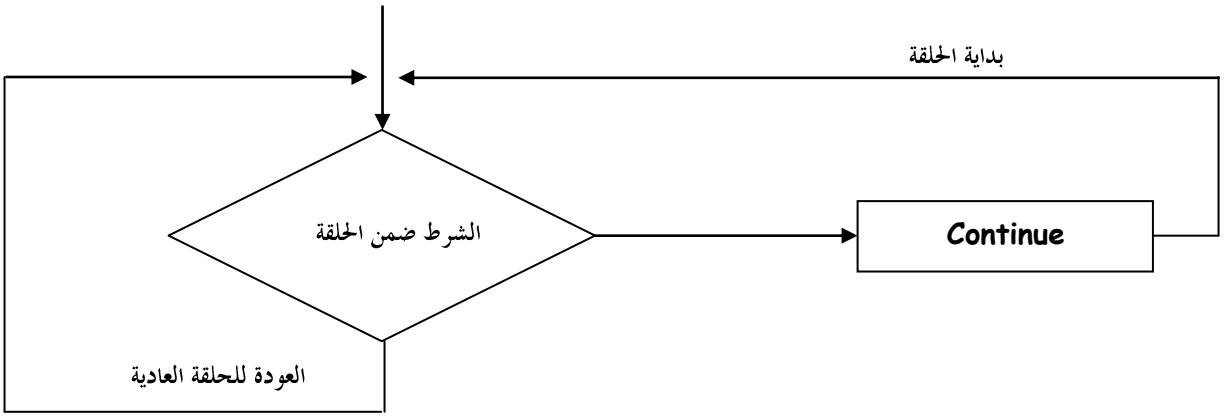
برنامج يطبع الأعداد من ١إلى ١٠ ويستخدم **Break** لإيقاف حلقة التكرار عند تحقق شرط معين

```
# include <iostream.h>
main()
{
int i;
for(i=1;i<10;++i)
{
cout<< i <<"\n";
if(i==10)break;//إيقاف الحلقة عندما تكون I=10
}
return 0;
}
```



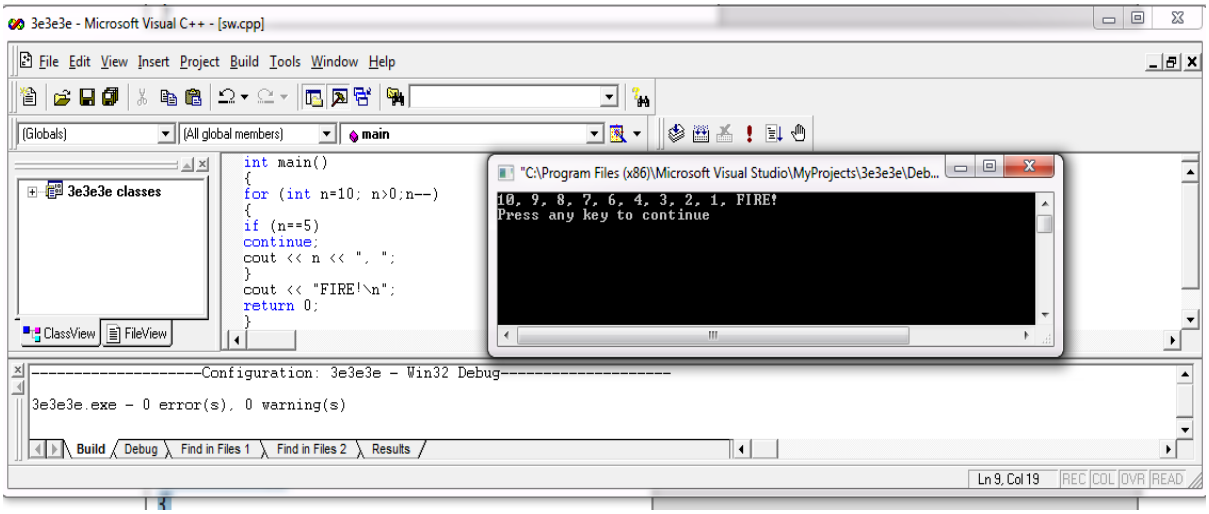
٢. التعليمة continue

تستخدم هذه التعليمة عندما نرغب بأن يتجاهل المترجم الجمل التي تلي التعليمة **continue** وإعادة التنفيذ إلى أعلى الحلقة.



طريقة عمل العبارة **continue**

برنامج يطبع الأعداد من ١ إلى ١٠ ماعدا الرقم ٥



٣. التعليمة Goto

بهذه التعليمة تستطيع إجبار المترجم على القفز من السطر الذي هو فيه إلى السطر الذي تريده ليكمل التنفيذ من المكان الذي أرسلته إليه ولاستخدام هذه التعليمة نحتاج لعلامة وهي النقطتين : لإستخدامها كدليل على السطر الذي سنرسل المترجم إليه .

برنامج يقوم بطباعة الأرقام أقل من ١٠ ويخرج رسالة كل مره ثم يعود لنقطة الإدخال بـــ GOTO

```
#include < iostream.h>
main ()
{
int x;
input: cin>>x;
if (x<10)
cout<<"value is under 10"<<endl;
goto input;
return 0;
}
```

الدوال(الوظائفFunction)

من الصعب في الــ++ C كتابة كل شي خلال البرنامج ، ولكن لغة الــ C++ وفرت لنا أشياء جاهزة للاستخدام نضع عناوينها فقط في البرنامج لأجل استثمار الكثير من الوقت. وكثير من الدوال والبيانات يتم تعريفها في أماكن أخرى خارج البرنامج كالمكتبة القياسية مثلا وأيضا كمكتبات لمشاريع خاصة تم تدوينها عن طريق مجموعة مبرمجين لغرض انجاز مشروع برمجي ما.

تعريف الدوال:

الدالة هى عبارة عن مقطع برمجي منفصل يؤدي عمل معين يمكن استدعائه من داخل البرنامج الرئيسى لتنفيذه وتكراره.

حيث أن هذا المقطع يكون موقعه ليس ضمن جسم الدالة الرئيسية وإنما خارج جسم الدالة الرئيسية **() main**، ويمكن باختصار تسمية الدالة (برنامج فرعي).

ومن فوائد الدوال ما يلي:-

١. تساعد الدوال المخزنة في ذاكرة الحاسوب على اختصار البرنامج إذ يُكتفي باستدعائها باسمها فقط لتقوم بالعمل المطلوب.
٢. تساعد الدوال المخزنة في مكتبةالحاسب على تلافي عمليات التكرار في خطوات البرنامج.
٣. تساعد الدوال الجاهزة على تسهيل عملية البرمجة نفسهاوسهولة تتبع الأخطاء و اختصار زمن البرمجة
٤. تنفيذ البرنامج بأسرع وقت ممكن.

تجزئة البرنامج modularity:

باستخدام الدوال يمكن تقسيم البرنامج الى برامج فرعية (روتين) صغيرة يسمى كل منه دالة ويتم استدعاء هذه الدوال خلال هيكل البرنامج كلما تحقق شرط معين أو للتفرع الى برنامج أخر، وهذا الأسلوب يُسهل تصميم البرنامج ويزيد سرعة تنفيذه .

الصيغة العامة للدالة

```
Type function_name (parameter1,parameter2, ....)
{
        statement 1;
        statement 2;
        statement 3;
        .
        .
        .
        return value
}
```

حيث أن:

Type:

هو النوع البياني الذي تعود به الدالة مثل (**int**او**float** او**long** او**void**..)

Function name:

وهو يشير إلى الاسم الذي سنطلقه على الدالة.

Parameter:

هو المتغير الذي نريد إرساله إلى الدالة ويمكن أن نرسل عدد غير محدد من المتغيرات حسب احتياجنا (هذه المتغيرات أو البارامترات تفيدنا في تنفيذ العمل المطلوب من الدالة) ويمكن أيضا ألا نرسل أى باراميتر أى أنها اختيارية.

Statement:

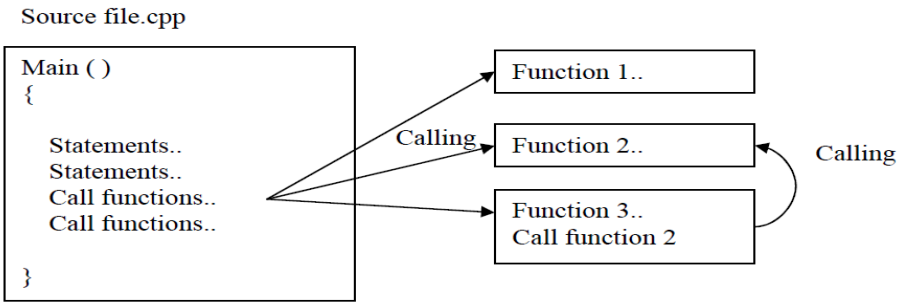
العبارات أو الجمل التي تُكون الدالة.

Return:

هى كلمة محجوزة للغة c++ وهى تفيد في إنهاء الدالة بصورة صحيحة وأيضا تقوم بإسناد القيمة التى بعدها وهى **Value** إلى اسم الدالة.

Value:

هى القيمة التي ستعيدها الدالة حيث أن كل دالة تعيد قيمة واحدة باسمها، ويمكن أن تكون هذه القيمة متغير أو ثابت أو رقم على شرط أن يكون من نفس النوع البياني الذي عرفت فيه الدالة أى من نفس نوع **Type** قبل اسم الدالة.



ملاحظات:

١. يمكن تعريف الدوال قبل الدالة الرئيسية **main**

٢. استدعاء الدوال يجب أن يكون من داخل الدالة الرئيسية

مثال:

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

٣. يمكن تعريف الدوال تحت الدالة الرئيسية **main** بشرط الإعلان عنها قبل الدالة

مثال:

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

شكل البرنامج بعد كتابته على الحاسب:

```
#include <iostream.h>

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

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

int get_sum (int a=0, int b=0)

أشكال الدوال:

-اجراء procedures

وهي دالة تقوم بعمل معين وتنفذه أو تطبعه على الشاشة ،وتسمى إجراء لأنها لا تعيد قيمة .

الصيغة العامة للإجراء:

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

يمكن أن يحتوي الإجراء أو الدالة على بارامترات 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:

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

الناتج: 15

-دالة Function

وهي دالة تقوم بعمل معين وتعيد قيمه .

الصيغة العامة للدالة:

- استخدام الكلمة المحجوزة return لإعادة قيمة:

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

أنواع الدوال :

١. **دوال قياسية** Standard Function

كل الدوال الجاهزة الموجودة في المكتبات مثل **Cout**و**pow(),Sum** ولا يمكن تعديلها .

٢. **دوال من تعريف المستخدم** User Defined Function

الدوال التي يقوم المستخدم بتكوينها ويستطيع التحكم فيها وتعديلها

تطبيق على الدوال القياسية

برنامج يستخدم دالة بسيطة لإيجاد مجموع عددين صحيحين

```
#include <iostream.h>
 int sum (int x,int y)// تعريف الدالة
{
 int z;
 z=x+y;
 return z;
}
int main()
{
 int c;
 c=sum(3,5);// استدعاء الدالة
 cout<<"3+5="<<c;
 return 0;
}
```

تطبيقات على الدوال من تعريف المستخدم

برنامجيقوم بطباعة جملة I'm a Function

```
// void function example
#include <iostream.h>

void printmessage ()          void هو نوع الدالة
{
 cout << "I'm a function!";
}

int main ()
{
 printmessage ();

}
```

```
I'm a function!
```

الدوال الخاليه Void function

Type تشير إلى النوع البياني الذي ستعود به الدالة ،ولكن لو أردنا ألا نعيد أى قيمة للدالة كما في بعض الدوال التى لا تحتاج أن نعيد قيمة لها أو أن الغرض من الدالة هو طباعة جملة مثلا ،فإذا أردنا جعل الدالة خالية نكتب كلمة **Void** مكان النوع **Type** وهذه الكلمة تعني فراغ أو لاشيء، بالإضافة إلى أن الدالة يجب ألا تحتوي على الكلمة **Return** لأنه ليس هناك قيمة س تعيدها وأخيرا ليس هناك حاجة لإسناد الدالة إلى متغير.

برنامج يقوم بقراءة عدد صحيح ثم يقوم بطباعة إذا كان الرقم زوجي أم فردي وذلك من خلال دالة من تعريف المستخدم اسمها **check**

```
#include <iostream.h>
int x;
check()
{
if ((x%2)==0)
cout<<"even"<<endl;  //   أطبع إذا كان الرقم زوجي
else

cout<<"odd"<<endl;// اطبع إذا كان الرقم فردي
}

main()
{
cin>>x;
check ();
return 0;
}
```

برنامج يقوم باستخراج أكبر رقم من بين رقمين مدخلين وذلك من خلال دالة من تعريف المستخدم اسمها **max**

```
#include < iostream.h>
int x,y;
max()
{
if (x>y)          //  للمقارنة بين العددين واختيار الأكبر
cout<<x<<endl;
else
cout<<y<<endl;
}
main( )
{
cin>>x>>y;        //هنا يتم إدخال القيمتين
max ( );
return 0;
}
```

المصفوفات Arrays

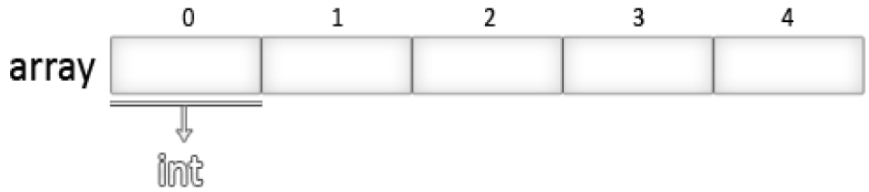
لنفرض أنه طلب منك كتابة برنامج بسيط لإدخال درجات عشرة طلاب؛ لكي تحل هذه المسألة باستخدام برنامج فإن عليك أن تقوم بالإعلان عن عدد كبير من المتغيرات العددية الصحيحة. وربما أن هذا مقبول نوعاً ما ؛ ولكن ماذا لو طلب منك إدخال درجات أكثر من **1000** طالب ،ولحل هذه المشكلة توفر لك لغة **C++** خاصية استخدام المصفوفات،صحيح انه يمكن حل مسائل من هذا النوع بدون استخدام المصفوفات لكن ماذا لو طلب منك البحث عن درجة طالب معين فلن يكون هناك أى حل إلا بواسطة المصفوفات.

تعريف المصفوفة:

هى عبارة عن مجموعة من البيانات التي تشترك في الاسم والنوع محفوظة في الذاكرة ولكنها تختلف في القيم المسندة إليها وتكون عناصر المصفوفة مرتبة بحيث يمكننا الوصول إلي أى عنصر نريده بتحديد ترتيبه في المصفوفة.

فمثلاً يمكن أن نخزن ٥ قيم من النوع **int** دون الحاجة للإعلان عن ٥ متغيرات بأسماء مختلفة وذلك باستخدام المصفوفات كما يلي:

```
int array[5] ;
```



المصفوفة array تحتوي ٥ قيم صحيحة

index	x
3	int
2	int
1	int
0	int

مميزات استخدام المصفوفات:

١. تقليل حجم البرنامج
٢. سهولة اسناد القيم واسترجاعها
٣. امكانية عمل البحث والترتيب
٤. سهولة الوصول لأي عنصر في المصفوفة

عيوب استخدام المصفوفات:

١. يمكن للمستخدم تحديد حجم المصفوفة عند تعريفها فقط
٢. يجب أن تحتوي جميع القيم على نوع من البيانات

والمصفوفات تنقسم لثلاثة أنواع من حيث أبعاد المصفوفة:

١. المصفوفات ذات البعد الواحد

٢. والمصفوفات ذات البعدين

٣. المصفوفات المتعددة الأبعاد

المصفوفات ذات البعد الواحد:

المصفوفة ذات البعد الواحد هى مجموعة من العناصر مرتبة بحيث يمكن الوصول إلى أى عنصر فيها باستخدام ترتيبه بالنسبة لأول عنصر في المصفوفة،حيث يأخذ أول عنصر الرقم صفر .

أولاً:الإعلان عن المصفوفة ذات البعد الواحد:

Type-specifier array-name [size];

حيث أن:

١. Type-specifierنوع المصفوفة (int – float -....) :

٢. Array – name اسم المصفوفة

٣. [Size]عدد عناصر المصفوفة

عدد عناصر المصفوفة يجب أن يكون بين قوسين[ ] مثال ; int mark [10]

عناصر المصفوفة:

المصفوفة السابقة تحوي هذه العناصر:

int mark[0] ; int mark[1] ; int mark[2] ;int mark[3] ; int mark[4] ; int mark[5] ;int mark[6] ; int mark[7] ; int mark[8] ;int mark[9] ;

وكما تلاحظ فإن المصفوفة مكونة من عشرة عناصر حسبما هو مكتوب في الإعلان السابق.

ويجب أن نلاحظ هنا أن أول عنصر في المصفوفة هو int mark[0]ووآخر عنصرهو:

وكما تلاحظ فإنه لا وجود للعنصر int mark[10] وهذا ما عليك أن تعرفه وهو بالغ الأهمية فللعد فيالمصفوفة يبدأ من العنصر رقم صفر وينتهى إلى العدد ما قبل الأخير من عدد عناصر المصفوفة المعلن عنه.

إدخال البيانات للمصفوفة

بإمكانك إدخال عناصر المصفوفة دون الحاجة إلى دالة من داخل برنامج فمثلاً بإمكانك كتابة السطر التالي:

int mark[7] = { 5,10,90,100,90,85,15};

وهذه الطريقة في حال أنك لا تريد أن يقوم المستخدم بإدخال أى أرقام للمصفوفةوبستخدام هذه الطريقة بإمكانك الاستغناء عن عدد عناصر المصفوفة الموجود بينقوسين هكذا:

int mark[] = { 5,10,90,100,90,85,15};

وسيقوم المترجم بحصر العناصر الموجودة في المصفوفةلكن ليس بإمكانك كتابة السطر السابق لكي تطلب من المستخدم إدخال عناصرالمصفوفة.

لاحظ: أنه في جميع طرق الإعلان عن المصفوفة فلابد عليك من تحديد حجم المصفوفة وإلا فإن المترجم سيعطيك رسالة خطأ.

أمثلة على إدخال البيانات للمصفوفة ذات بعد واحد

تعريف مصفوفة وإسناد كل بياناتها في نفس الوقت:
بقية الخانات ستكون صفرية:
كل الخانات ستكون أصفار:

```
1. int x[5] = { 1, 7, 10, 2, 5};
2. int y[10] = {3, 5};
3. int z[4] = {0};
```

تعريف مصفوفة:
إسناد قيمة للخانة الأولى في المصفوفة:
إسناد قيمة للخانة الثانية في المصفوفة:
إسناد قيمة للخانة الثالثة في المصفوفة:

```
1. int a[3];
2. a[0] = 1;
3. a[1] = 7;
4. a[2] = 10;
```

10
7
1

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

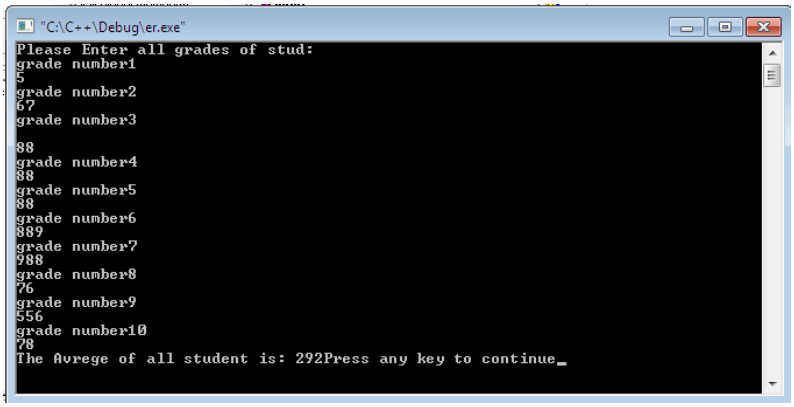
إدخال جميع قيم المصفوفة باستخدام دالة For

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

تطبيقات على المصفوفات

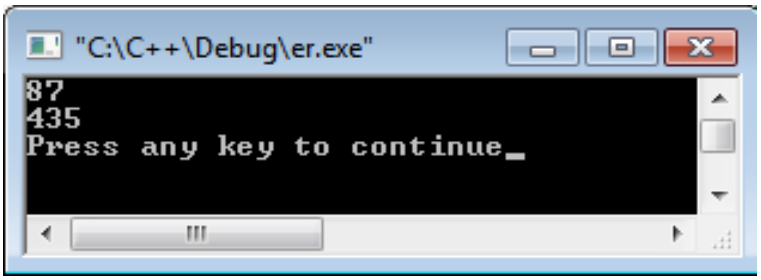
برنامج يحسب متوسط درجات ١٠ طلاب باستخدام المصفوفات(مصفوفة ذات بعد واحد)

```
//برنامج يحسب متوسط درجات 10 طلاب باستخدام المصفوفات
#include <iostream.h>
main ( )
{
int stud[10] ,total=0 , i ;
float Avrege;
cout << "Please Enter all grades of stud:\n" ;
for (i=0 ; i<10 ; i++)
{
cout << "grade number" << i+1 << endl;
cin >> stud[i] ;
total=total+stud[ i ] ;
}
Avrege=total /10;
cout << "The Avrege of all student is: " << Avrege ;
return 0;}
```



برنامج يحسب المجموع والمتوسط لمصفوفة بإدخال عناصر المصفوفة مباشرة

```
#include < iostream.h>
int m,i;
main ()
{
int a[5]={87,67,81,90,55};
int s=0;
for(i=0;i<5;i++)
s=s+m[a];
float avg=s/5;
cout<<avg<<endl;
cout<<s<<endl;
return 0;
}
```



برنامج لإدخال مرتبات عشرة موظفين ثم يقوم بحساب متوسط المرتبات(مصفوفة ذات بعد واحد)

```
#include<iostream.h>
#define max 10
void main()
{
float salary[max];
float average,sum;
int count;
sum=0.0;
for(count=0;count<max;count++)
{
cout<<"please enter salary for employee\n";
cin>>salary[count];
sum=sum+salary[count];
}
average=sum/max;
cout<<"\n salary average is ="<<average;
}
```

المصفوفات ذات البعدين(ثنائية البعد) :

المصفوفة ذات البعدين تحتوي على عناصر من نفس النوع، ولكنها مرتبة في صفوف وأعمدة.وبالتالي تختلف طريقة الوصول للعناصر إذ يلزم لتحديد العنصر استخدام رقم الصف ورقم العمود.

الشكل التالي يوضح مصفوفة ذات بعدين:

$$B=\begin{pmatrix} 12 & 23 & 15 \\ 25 & 35 & 89 \\ 90 & 80 & 16 \end{pmatrix}$$

ثانياً:الإعلان عن المصفوفة ذات البعدين:

يتم الإعلان عن هذه المصفوفة كالتالي:

name [index 1][index 2];_Type-specifier array

وعناصر المصفوفة في هذه الحالة كما ذكرنا تحدد باستخدام رقمين هما رقم الصف ورقم العمود، فالعنصر ١٢ يقع في العمود ا لأول والصف الأول .لاحظ أن الترقيم في المصفوفة يبدأ بالرقم صفر دائما.

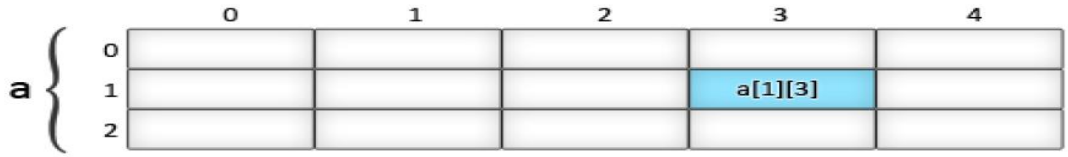
B [0] [0] =12

أمثلة على إدخال البيانات للمصفوفة ذات بعدين

```
int a[3][5];
```

وللوصول إلى العنصر الواقع في السطر الثاني والعمود الرابع نكتب

```
a[1][3]
```



برنامج مصفوفة تتكون من خمسة أعمدة وثلاثة صفوف(ذات بعدين)

```
#include <iostream.h>
main ()
{
int m[5][3];
int i,j;
for(i=0;i<5;i++)
for(j=0;j>3;j++)
cin>>m[i][j];
return 0;
}
```

المصفوفات المتعددة Multidimensional Array:

يمكن للمصفوفات في C++ أن تكون متعددة الأبعاد ويمكن كذلك أن يكون كل بعد بحجم مختلف، الاستعمال الشائع للمصفوفات متعددة الأبعاد هو تمثيل الجداول Tables التالي تحتوي على بيانات مرتبة في صورة صفوف وأعمدة ولتمثيل الجدول نحتاج لبعدين الأول يمثل الصفوف والثاني يمثل الأعمدة.

الشكل التالي يبين مصفوفة **A** تحتوى على ثلاثة صفوف وأربع أعمدة.

	Column 0	Column1	Column2	Column 3
Row 0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
Row 1	A[1][0]	A[1][1]	A[1][2]	A[1][3]
Row 2	A[2][0]	A[2][1]	A[2][2]	A[2][3]

يتم تمثيل أي عنصر في المصفوفة **A** على الصورة **A[i][j]** حيث:-

A : اسم المصفوفة.

i : رقم الصف الذي ينتمي إليه العنصر.

j : رقم العمود الذي ينتمي إليه العنصر.

لاحظ أن كل العناصر الموجودة في الصف الأول مثلاً يكون الفهرس الأول لها هو **0** وكل العناصر الموجودة في العمود الرابع يكون الفهرس الثاني لها هو **3**.

ثلاثية الأبعاد

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

رباعية الأبعاد

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

وهكذا يمكن إضافة أبعاد للمصفوفة حسب الحاجة:

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

يمكن تمهيد قيمة المصفوفة المتعددة الأبعاد عند الإعلان عنها وذلك كالآتي:

intB[2][2]={{1,2},{3,4}};

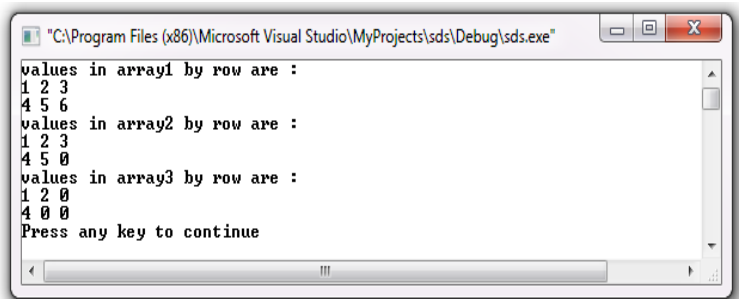
حيث: **b[1][1]=4, b[1][0]=3, b[0][1]=2, b[0][0]=1**

ملاحظة:

المصفوفة متعددة الأبعاد إذا تم تمهيدها عند قيم لا يتوافق عددها مع حجم المصفوفة فإن المترجم سيملأ بقية العناصر أصفار.

البرنامج التالي يوضح كيفية تمهيد مصفوفات متعددة الأبعاد عند الإعلان عنها:

```
// initializing multidimensional arrays
#include<iostream.h>
void printArray(int [ ] [3]);
int main( )
//continued
{
int   array1[2] [3] = { {1, 2, 3}, {4, 5, 6}},
       array2[2] [3] = {1, 2, 3, 4, 5},
       array3[2] [3] = { {1, 2}, {4} };
cout << "values in array1 by row are : " << endl;
printArray(array1);
//Continued
cout << "values in array2 by row are : " << endl;
printArray(array2);
cout << "values in array3 by row are : " << endl;
printArray(array3);
return 0;
}
void printArray(int a[ ][3])
{
for (int i=0; i<2; i++) {
  for (int j=0; j<3; j++)
cout << a[i][j] <<' ';
cout << endl;
    }
    }
```



هياكل البيانات في لغة Data structure in C++

التركيب Struct

هو مجموعة من المتغيرات المختلفة في النوع يتم التعامل معها كوحدة واحدة (هيكل) وتمثل هياكل البيانات طريقة من طرق هيكلة البرمجة حيث أنها تساعد في تنظيم وتسهيل كتابة البرنامج .

كيفية الإعلان عن التركيب Structutre

للإعلان عن التركيب نستخدم الكلمة المحجوزة **Struct** وهي اختصار لكلمة **Structure** ومعناها تركيب.

الشكل العام لكيفية الإعلان عن متغير

```
#include<iostream.h>
structstruct-name اسم التركيب
{
……..;
……..;
……..;
……..;

};          فاصلة منقوطة في نهاية التركيب
Void main ( )
{


}
```

ملاحظات :

١. يكتب التركيب قبل الدالة الرئيسية .
٢. البيانات مختلفة في النوع بعضها من نوع **int** وبعضها **Char** وبعضها **Float**.....الخ
٣. فمثلاً لكتابة برنامج لتخزين بيانات موظفين في الشركة نحتاج لتخزين ما يلي:
   i. اسم الموظف وهو متغير من نوع مصفوفة حرفية **char name[40]**
   ii. عنوان الموظف وهو متغير من نوع مصفوفة حرفية **char address [40]**
   iii. عمر الموظف وهو متغير من نوع عدد صحيح **int age**
   iv. مرتب الموظف وهو متغير من نوع عدد حقيقي **flaot salary**
٤. جميع هذه البيانات يجب التعامل معها كوحدة واحدة لأنها لموظف واحد ولذلك فإننا بحاجة لسجل خاص بهذا الموظف.
٥. يتم وضع جميع بيانات التركيب بين قوسين
٦. يختم الإعلان عن التركيب بفاصلة منقوطة بعد القوس الثاني

مثال لتخزين بيانات موظف في صورة تركيب

Struct:

num	name	age	address	phone
4	20	4	8	4

= 44 byte

```
1.  struct students
2.  {
3.      int num;
4.      char name[20];
5.      int age;
6.      string address;
7.      double phone;
8.  };
9.  void main()
10. {
11.
12. }
```

رقم
اسم
عمر
عنوان
تلفون

طرق اسناد القيم للتركيب :

١. إسناد القيم دفعة واحدة

مثال

```
Students  stud={1,"ali",25,"Cairo",01111111111};
```

٢. إسناد كل قيمة بشكل مستقل

مثال

```
1. students  stud;
2. stud.num=1;
3. stud.name="ali";
4. stud.age=25;
5. stud.address="Cairo";
6. stud.phone=01111111111
```

٣. استخدام الدالة **cin** لطلب البيانات من المستخدم (أثناء التشغيل)

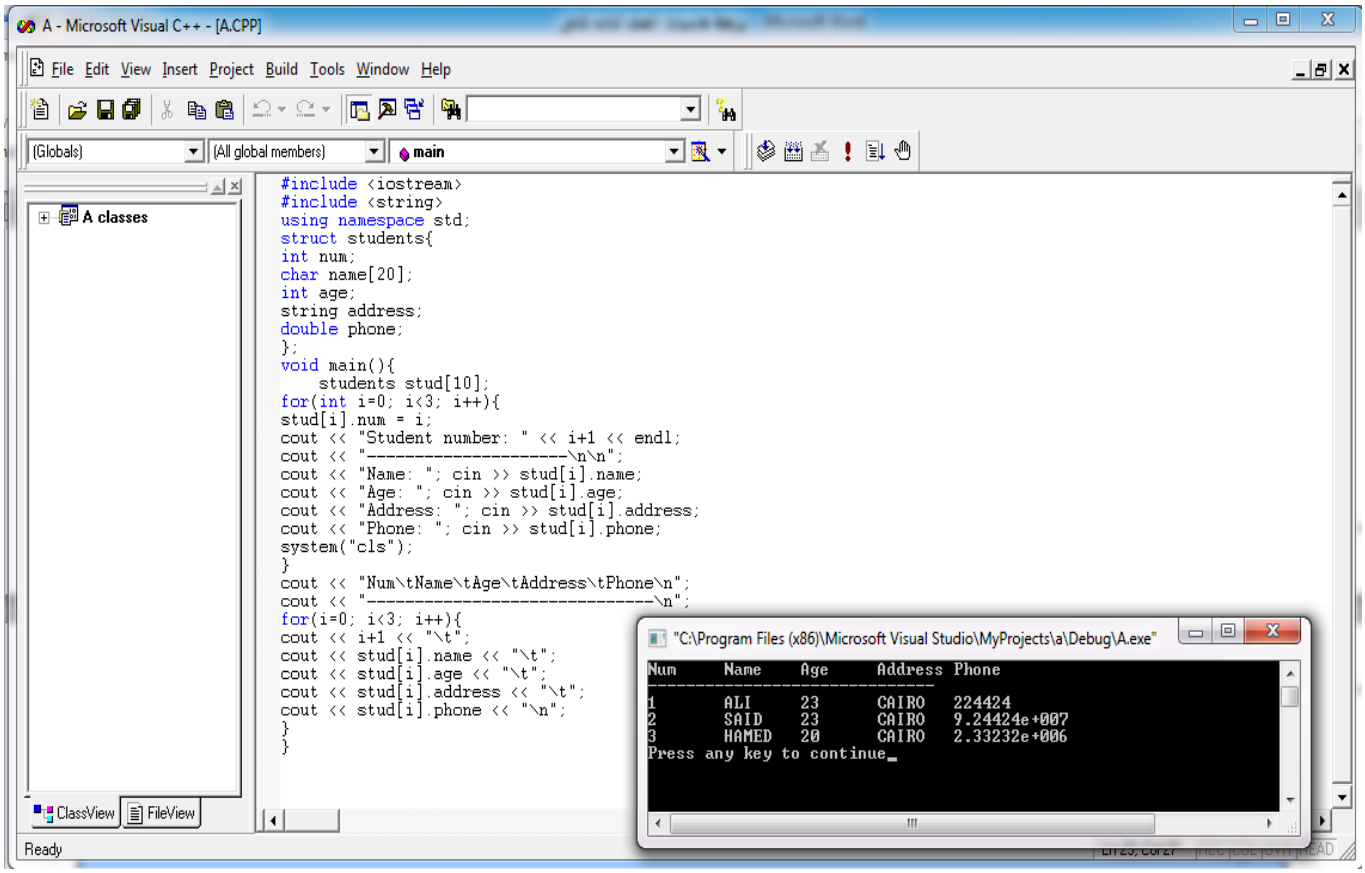
مثال

```
1.  students stud;
2.  cout<< "Enter number: ":      cin >> stud.num;
3.  cout<< "Enter name: ":        cin >> stud.name;
4.  cout<< "Enter age: ":         cin >> stud.age;
5.  cout<< "Enter address: ":     cin >> stud.address;
6.  cout<< "Enter phone: ":       cin >> stud.phone;
```

برنامج يستقبل معلومات عن اسم الموظف وعنوانه وعمره وراتبه باستخدام التراكيب Structures

```
#include<iostream.h>
struct employee
{
char name[40];
char address[40];
int age;
float salary;
};
void main()
{
struct employee emp;
cout<<"enter name"<<endl;
cin>>emp.name;
cout<<"enter address"<<endl;
cin>>emp.address;
cout<<"enter age"<<endl;
cin>>emp.age;
cout<<"enter salary"<<endl;
cin>>emp.salary;
}
```

برنامج لتسجيل بيانات الطلاب وعرضها بشكل منسق باستخدام التراكيب Structures



استخدام المؤشرات Pointers والمراجع Refrences

مثال توضيحي:

لنفترض أنك وأربعة من أصدقاءك ذهبتم إلى المطعم فإنكم ستحجزون طاولة بشكل عشوائي ،وبفرض أن الطاولة تستوعب أربعة كراسي فقط فإن أي طاولة ستكون مناسبة لجلوس أربعة أشخاص فقط(وكذلك المتغير يفعل(لكنه يحجز ٨ بت على الأقل) ولنفرض أن زملاءً لك جاءوا إلى المطعم بالصدفة وأردت أن يتناولوا الطعام معك فإن هذا غير ممكن لعدم حجز طاولة مسبقاً، كما أن حجز طاولة احتياطياً يعتبر إهداراً في حالة عدم حضور أصدقاءك ،ولحل المشكلة يجب أن تتفق مع مدير المطعم على ضم طاولة إضافية في حالة حضور أصدقاء آخرين وهو ماتفعله المؤشرات.

تعريف المؤشرات Pointers :

– عبارة عن متغيرات في الذاكرة تشير إلى عناوين والتي بدورها تشير إلى قيم .

– عبارة عن مصفوفة مفتوحة لحجز مواقع عشوائية في الذاكرة حسب الحاجة.

يستخدم المؤشر في لغة C++ كعنوان لمتغير في الذاكرة ، أحد الاستعمالات المهمة للمؤشرات هو التخصيص الديناميكي للذاكرة حيث يتم استعمال المؤشرات لإنشاء بنية بيانات لتخزين البيانات في الذاكرة.يتم الإعلان عن المؤشرات قبل استخدامها في البرنامج فمثلاً العبارة :

```
int *Countptr;
```

تعلن عن مؤشر **Countptr** ليشير إلى متغير من النوع **int** (* المذكورة قبل اسم المؤشر تشير لذلك) وكل متغير يعلن عنه كمؤشر يجب أن يكتب في الإعلان مسبوقاً بـ* فمثلاً الإعلان :**Float *xptr, *yptr;**

يشير لأن كلاً من **xptr** و**yptr** موقعي مؤشرات لقيم من النوع **Float** ويمكن أن تستخدم المؤشرات لتشير لأي نوع بيانات آخر.

تذكر دائماً عند الإعلان عن أي مؤشر أن تسبق * كل مؤشر على حدة فمثلاً الإعلان :

Int *xptr, yptr; ليس صحيحاً.

يجب أن تعلن عن هذه المؤشرات كالآتي: **int *xptr, *yptr;**

مكن تمهيد المؤشرات عند الإعلان عنها عند قيمة **0** أو **null** أو عند قيمة عنوان في الذاكرة . المؤشر الذي يحمل القيمة **0** أو **null** لا يشير لأي متغير . تمهيد المؤشر عند **0** يكافئ تمهيده عند **null** ولكن في+ +C يفضل تمهيد المؤشر عند القيمة **0**.

معاملات المؤشرات :

١–معامل العنوان (&)تستخدم مع المؤشرات لمعرفة عناوينها

فعندما نعلن عن متغير فإنه سيخزن في إحدى الخلايا المتاحة في الذاكرة بشكل آلي من قبل المترجم ونظام الشغيل ،فإذا أردنا معرفة مكان تخزين هذا المتغير نسبق اسم المتغير بالرمز &.

فمثلاً الأمر: **a=&b** ستضع عنوان المتغير **b** في المتغير **a**

و بالتالي فإن المتغير الذي يحوي عنوان متغير آخر يسمى مؤشر.

٢– معامل المرجع (*) تستخدم مع المؤشرات لمعرفة قيمها .

إذا سبقنا المؤشر **a** بمعامل المرجع* أي إذا كتبنا **a*** فإن العبارة **c=*a;** تعني ضع القيمة التي يشير إليها المؤشر **a** في المتغير **C**

كيفية الإعلان عن المؤشرات :

Type * Pointer_ name;

Type: هو نوع البيانات التي يشير إليها المؤشر وليس نوع المؤشر نفسه

Pointer_name: هو اسم المؤشر

أمثلة:

int * number;
char *charcter;
float *greatnumber;

لدينا في الأمثله السابقة ثلاثة مؤشرات منها يشير إلى نوع مختلف من البيانات ومع ذلك فإن كل واحد من هذه المؤشرات يحجز نفس الحجم في الذاكرة بحسب نظام التشغيل أما البيانات التي تشير إليها هذه المؤشرات فإن لها حجوم مختلفة .

تطبيق على استخدام المؤشرات

```
// my first pointer
#include <iostream>
using namespace std;
int main ()
{
  int a = 5, b = 15;
  int *ptr;
   ptr= &a;
  *ptr = 10;
   ptr = &b;
  *ptr = 20;
  cout<<"a="<<a<<" , b="<<b;
  return 0;
}
```

```
a=10 , b=20
```

- ✓ لاحظ أنه تم تعديل قيم a و b بشكل غير مباشر حيث جعلنا المؤشّر ptr يشير إلى a عن طريق الرمز & ثمّ أسندنا القيمة 10 إلى مايشير إليه المؤشّر ptr وكذلك الأمر بالنسبة لـ b.
- ✓ كما يمكن أن يأخذ المؤشّر خلال البرنامج عناوين مختلفة حيث استخدمنا المؤشّر ptr ليشير إلى a ثمّ إلى b.

المؤشرات والمصفوفات:

المؤشرات	المصفوفات
يوضع في الذاكرة عنوان أول عنصر من عناصرها	يوضع في الذاكرة عنوان أول عنصر من عناصرها
يتم حجز موقع عشوائي في الذاكرة	يتم حجز موقع عشوائي في الذاكرة
يتم الإعلان عن المؤشرات كما يلي: **int * p=&X;** حيث يأخذ المؤشر **P** عنوان المتغير **X** وبالتالي تصبح قيمة **P** هي عنوان المتغير **X**، وبالتالي نستطيع أن نعطي قيم مختلفة للمؤشر **P**.	يتم الإعلان عن مصفوفة بها ثلاث عناصر كما يلي: **int X[٣];** والتي تقوم بحجز ثلاث مواقع عشوائية في الذاكرة ووضع كل قيمة فيما يقابلها من مواقع: مثال: **X[0]=9,X[1]=8,X[2]=7**

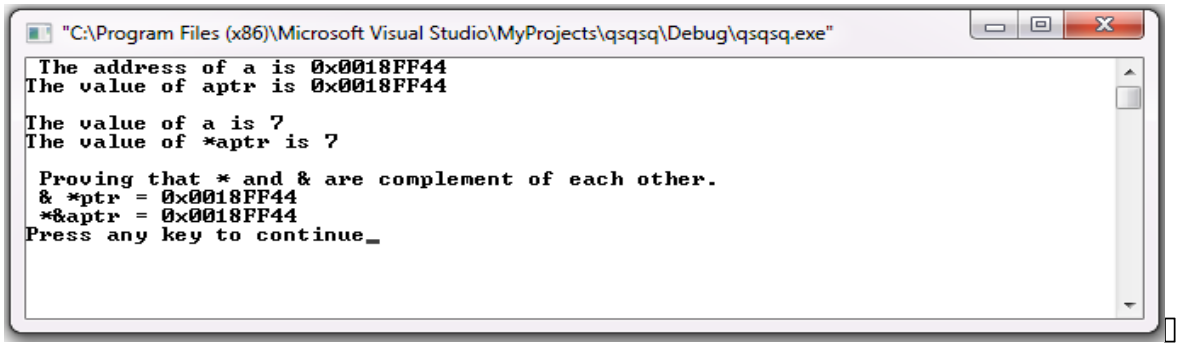
تطبيق١ على المؤشرات والمصفوفات:

```
// more pointers
#include <iostream>
using namespace std;
int main ()
{
  int numbers[5];
  int * p;
  p = numbers;  *p = 10;
  p++;  *p = 20;
  p = &numbers[2];  *p = 30;
  p = numbers + 3;  *p = 40;
  p = numbers;  *(p+4) = 50;
  for (int n=0; n<5; n++)
    cout << numbers[n] << ", ";
  return 0;
}
```

```
10, 20, 30, 40, 50,
```

تطبيق ٢ على المؤشرات والمصفوفات:

```
// using the & and * operators
#include<iostream.h>
main ( )
{
  int a ;                              //a is an integer
  int  *aptr;                        // aptr is apointer to an integer
   a = 7;
   aptr = &a;                    // aptr set to address of a
   cout <<" The address of a is " << &a <<endl
            << "The value of aptr is " << aptr<< endl<< endl;

  cout << "The value of a is " << a<< endl
       << "The value of *aptr is " << *aptr<< endl<<endl;
  cout<<" Proving that * and & are complement of "
         << "each other." <<endl<< " & *ptr = "<< & *aptr
         << endl<< " *&aptr = " << *&aptr <<endl;
   return  0;
}
```



جدول يوضح مميزات وعيوب المتغير /المصفوفه/ التركيب /المؤشر

النوع	الطريقة	المميزات	العيوب
المتغير	يحجز موقع للقيمة	سهل التعريف والاستدعاء والإسناد.	لكل قيمة يجب تعريف متغير جديد يصعب الحصول على المتغيرات فكل متغير يحتاج إلى استدعاءه باسمه.
المصفوفة	تحجز عدة مواقع متجاورة متساوية الطول والنوع	توفر طريقة سهلة لإدخال القيم واستخراجها بدالة دوران مثل for	يجب تحديد حجم المصفوفة مسبقاً ولا يمكن تحديد حجمها أثناء التشغيل. كل متغيرات المصفوفة الواحدة يجب أن تكون من نفس النوع لا يمكن زيادة أو تقليص حجمها بحسب البيانات الفعلية.
التركيب	يحجز عدة مواقع متجاورة غير متساوية الطول أو النوع حسب الرغبة	نفس مميزات المصفوفة يمكن عمل تركيب داخل تركيب توفر طريقة سهلة لتجميع متغيرات مختلفة في تركيب واحد يسهل التعامل معه يمكن عمل مصفوفة من التركيب	تبقى مشكلة مصفوفة التركيب حيث لا يمكن زيادة أو تقليص حجمها لا يمكن تحديد حجم المصفوفة أثناء التشغيل.
المؤشر	يحجز عنوان ويضع فيه قيمة، ويستطيع التحرك لوضع قيمة في الموقع المجاور أو استدعاء قيمة منه.	يستطيع التحرك في الذاكرة وتعبئتها بالقيم حسب الطلب. لا يحجز إلا المساحة المطلوبة ويمكن زيادة حجمه وتقليصه يضيف إمكانات هائلة للمصفوفات والتراكيب وحتى الكائنات يمكن تعريف مؤشر من نوع مصفوفة وبالتالي يمكن تعيين حجمها أثناء التشغيل.	صعوبة المؤشرات فهي تتعامل مع العناوين وليس القيم خطورة التعامل مع المؤشرات صعوبة اكتشاف أخطاء المؤشرات تستطيع تحديد حجم المصفوفة من نوع مؤشر أثناء التشغيل لكن بعد تحديده لا يمكن زيادته أو تقليصه.

طريقة حذف المؤشر:

```
int  *P=&x;
```

يتم اسناد القيمة"صفر" للمؤشر:

```
P=0;
```

أسئلة على الباب الأول

١. أكمل :

i. كل برنامج في C++ يبدأ من الدالة .....

ii. كل عبارة في C++ تنتهي بـــ .....

iii. الرمز // يدل على أن هذا السطر .....

iv. التعليمة return(0) .....

v. المتغيرات هي .....

٢. ضع علامة ()أو (×)

i. المتغير NUm هو نفسه المتغير num في لغة C++ ( )

ii. اسم المتغير Unsigned int a=-3.4 يعد صحيح ( )

iii. اسم المتغير Unsigned int a=-3.4; يعد صحيح ( )

iv. اسم المتغير int 4men; يعد صحيح ( )

v. اسم المتغير double d='s; يعد صحيح ( )

vi. اسم المتغير short hi ; يعد صحيح ( )

vii. يجب الإعلان عن المتغيرات قبل استعمالها في البرنامج( )

viii. يجب تحديد نوع المتغيرات عند الإعلان عنها( )

ix. لا تفرق C++ بين المتغيرات Number و number( )

x. تمتلك العوامل الحسابية + ، – و % نفس درجة الأولوية ( )

xi. برنامج C++ والذي يقوم بطباعة ثلاث أسطر على الشاشة يجب أن يحتوى على ثلاث عبارات تستعمل**cout**( )

٣. ما الذي يطبعه الأمر التالي :

```
cout<<"*\n**\n***\n****\n***** ";
```

٤. حدد أولوية تنفيذ العمليات الحسابية التالية :

```
a) x = 7 + 3 * 6 / 2 – 1;
b) x = 2 % 2 + 2 * 2 – 2 / 2;
c) x = ( 3 * 9 * ( 3 + ( 9 * 3 / ( 3 ) ) ) );
```

٥. أكتب عبارة **C++** صحيحة تقوم بالآتي:

i. تعريف المتغيرات x، y ، z و result لتكون من النوع int.

ii. الطلب من المستخدم إدخال ثلاثة أرقام صحيحة.

٦. أي من العبارات الآتية تعطي المخرجات التالية:

1 2
3 4

```
cout<< " 1\t2\t\n3\t4\n";
cout <<'1' << '\t' << '2' << '\n' <<'3' <<'\t' <<'4' <<'\n';
cout<< "1 \n 2\t 3\n 4\t";
cout <<1 << '\t' << 2 << '\n' <<3 <<'\t' <<4 <<'\n';
```

البـــاب الثاني

البرمجة كائنية التوجه Object Oriented Programming(OOP)

أنـواع البــرمجــة:

١- البرمجة الهيكلية:

وتعتمد على طريقتين:

١. طريقة البرمجة المتتالية:حيث أن المبرمج يقوم ببرمجة ما يخصه ثم ينتقل البرنامج إلى مبرمج آخر ليقوم بدوره في ما يخصه وهكذا.

٢. طريقة البرمجة المستقلة :حيث يعمل كل مبرمج على جزء من البرنامج ثم يتم تجميع الأجزاء في برنامج واحد.

عيوبها:

١. التأخير في تنفيذ البرنامج
٢. كل مبرمج يجب أن يفهم ما عمله زميله
٣. أي خطأ في دالة يوقف البرنامج
٤. أي تعديل في برنامج فرعي يحتاج لتعديل في البرنامج الرئيسي

٢- البرمجة كائنية التوجهOOP:

يطلق عليها أيضاً البرمجة الشيئية أو البرمجة الموجهه نحو الهدف وتتسم بـــ:

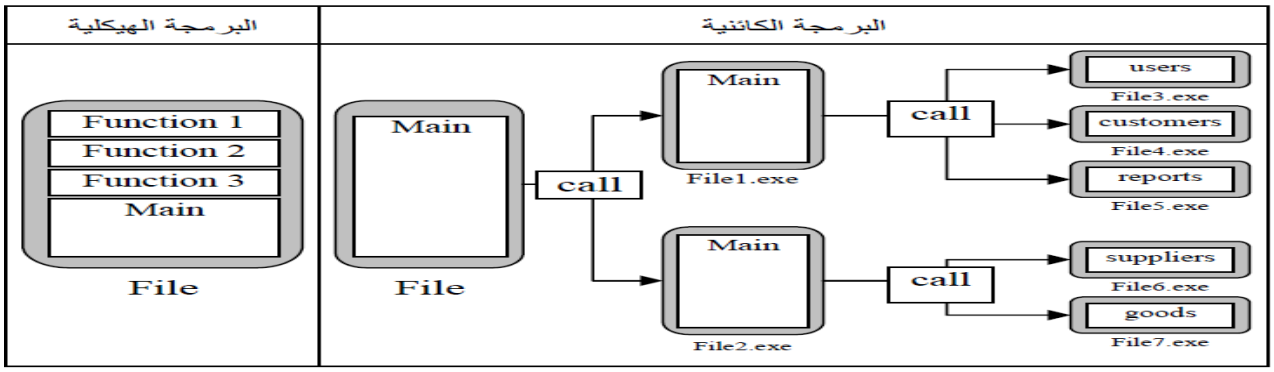
١. وهي طريقة لتجميع البيانات والوظائف معاً في قالب واحد يسمة الفئه **Class**.
٢. أجزاء البرنامج عبارة عن برامج فرعية مستقلة
٣. كل برنامج فرعي ينفذه مبرمج أو فريق
٤. إذا كبِر البرنامج الفرعي فيمكن تقسيمه إلى برامج فرعيه عن طريق التوريث

مميزاتها:

١. كل المبرمجين يعملون في نفس الوقت
٢. يمكن لكل مبرمج أن يبرمج بأي لغة
٣. كل برنامج له حماية لبياناته
٤. التعديل في البرنامج الفرعي لا يحتاج إلى التعديل في البرنامج الرئيسي
٥. إذا تعطل برنامج فرعي فلا يتوقف البرنامج الرئيسي.

مقارنة بين التركيب **Struct**(البرمجة الهيكلية)والفئه **Class**(البرمجة كائنية التوجه)	
التركيب **Struct**	الفئه **Class**
عدم القدرة على إخفاء البيانات	له القدرة على اخفاء البيانات
الوصول للمتغيرات متاح للجميع	يوجد مستويات حماية للمتغيرات
الوصول للبيانات داخل التركيب يتم بواسطة متغيرات التركيب	الوصول إلى البيانات داخل الفئة يتم بواسطة "خصائص" دوال الكائن

شكل يوضح مقارنة بين أنواع البرمجة



- الفرق بين البرمجة الشيئية والبرمجة التقليدية :

يختلف البرنامج المكتوب بأسلوب oop عن البرنامج المكتوب بالطريقة التقليدية :-

1. **وحدة بناء البرنامج** : كان البرنامج التقليدي عبارة عن دوال رئيسية ومجموعة دوال فرعية ويتم استدعاء الدوال الفرعية من داخل الدالة الرئيسية حسب تسلسل البرنامج ، أما في البرنامج المكتوب بأسلوب oop فقد أصبحت وحدة بناء البرنامج فئة (Class) التي يتكون من البيانات والدوال التي تعمل على هذه البيانات ويتم استدعاء الدوال كعناصر للفصائل على البيانات حسب فكرة البرنامج .
2. **النظرة إلى البيانات** : في البرنامج التقليدي كان مجهود المبرمج ينص على كود البرنامج أي سطور البرنامج التي تتولى سير العمليات بينما كان النظرة إلى البيانات نظرة ثانوية ، **ولكن** في أسلوب برمجة oop فتعتبر البيانات جزء مهم من البرنامج وبالتالي أصبحت لدينا مكتبة من الفصائل التي تحتوي على البيانات والدوال التي تتعامل معها .

مفاهيم البرمجة كائنية التوجه:

١. الكائن Object

مثل الأزرار والنوفذ وصناديق الحوار

٢. الخصائص Properties

والتي تحدد شكل وسمة الكائن (لونه وحجمه)

٣. الوسائل method

سلوك أو أفعال مصاحبه للكائن

٤. الحدث Event

الفعل الذي يقع على الكائن مثل الضغط بالفأرة

٥. الفئه Class

هو عبارة عن بناء برمجي يحتوي مجموعة من الكائنات التي تشترك في الخصائص

٦. المستوى العام public

كلمة مفتاحية تسمح للوصول لكل عنصر في الفئه ولكل المستخدمين

٧. المستوى الخاص Private

كلمة مفتاحية تسمح الوصول لعناصر معينه في الفئه

٨. المستوى المحمي Protected

كلمة مفتاحية تسمح لإشتقاق عناصر وتوريثها لأصناف أخرى.

- والإعلان عن الفصيلة يتكون من :-

1) أولاً الكلمة المحجوزة (Class) يليها اسم الفصيلة (ClassName) ويخضع لقواعد الفصيلة هي :-

أولاً : أن يكون اسم الفصيلة أحد الكلمات المحجوزة أو الكلمات التي تحمل معنى خاص مثل main .

ثانياً : يمكن أن يحتوي الاسم على أي حرف من الحروف الأبجدية (a - z) سواء كانت صغيرة أو كبيرة أو أي رقم من الأرقام (9 - 0) كما يمكن أن يحتوي الشرطة السفلى (_) ولكن أن لا يبدأ برقم .

ثالثاً : لا قيود على طول الاسم ، وتتيح هذه الميزة باستخدام أسماء معبرة عن مضمونها ، ومن الأفضل دائماً استخدام الاسم المعبر عن محتوى الفصيلة لتسهيل التعامل مع الفصائل .

رابعاً : الحروف الكبيرة والصغيرة ليست متكافئة في C++ فمثلاً اسم الصنف My_CLASS يختلف عن my_class وكلاهما يختلف عن MY_class .

2) تحديد درجة الحماية : وتبدأ عادة بدرجة الحماية الخاصة Private وتلي هذه الكلمة البيانات والدوال الخاصة بالفصيلة .

3) تحديد درجة الحماية الأخرى : مثل Public وتوجد ثلاثة أنواع من درجات الحماية كما في الجدول التالي :

درجة الحماية	المعنى
Private	تعني أن البيانات خاصة بهذه الفصيلة ولا يمكن الوصول إليها إلا بواسطة هذه الفصيلة.
Public	تعني أن البيانات التي تليها عامة ، ويمكن لأي دالة الوصول إليها واستعمالها .
Protected	تفيد في حالة توريث الفصيلة ، حيث يسمح بالفصائل التي ورثت باستعمال أعضاء الفصلية الأساسية .

ملاحظات:

١. يفضل أن يكتب اسم الفئه بحرف كابيتال

٢. تكتب الفئة class قبل الدالة الرئيسية

٣. كل ما يندرج تحت المستوى Public يمكن استخدامه داخل الفئه وخارجها

٤. المتغيرات التي تم الإعلان عنها في الجزء Private لا يمكن استخدامها خارج الفئه

٥. في حالة عدم ذكر Public أو Private فإن الكلمة الإفتراضيه هي Private

```
1.  Class class_name                          كلمة مفتاحية class واسم الفئة
2.  {
3.    public:                                 منطقة الصفات والخصائص العامة
4.        data type  member1;                 تعريف متغير/ صفة عامة
5.        data type function_name(per)        تعريف دالة / خاصية عامة
6.        ….  …..  …..
7.    Private:                                منطقة الصفات والخصائص المحمية
8.        data type  member1;                 تعريف متغير/ صفة محمية
9.        data type function_name(per)        تعريف دالة / خاصية محمية
10.       ….  …..  …..
11.   Protected:                              منطقة الصفات والخصائص الموروثة
12.       data type  member1;                 تعريف متغير/ صفة يمكن توريثها
13.       data type function_name(per)        تعريف دالة / خاصية يمكن توريثها
14.       ….  …..  …..
15. };
```

- باستخدام الفصائل class اكتب برنامج ليحسب مساحة ومحيط الدائرة علماً بأن المساحة = πr^2 والمحيط = $2 \pi r$ $\pi = 3.14$

```
#include <iostream.h>
class circle
{
private :
      float r;
      double area1,circum1;

public :
      void area(float r1)
      {
            r=r1;
            area1=3.14*r*r;
            cout<<"the area = "<<area1<<endl;
      }
      void circum()
      {
            circum1=2*3.14*r;
            cout<<"circle circum = "<<circum1<<endl;
      }
};
void main()
{
      circle ob1;
      float n ;
      cin>>n;
      ob1.area(n);
      ob1.circum ();
}
```

النتائج ○

```
10
the area = 314
circle circum = 62.8
Press any key to continue
```

مبادىء البرمجة كائنية التوجه :

١. مبدأ التغليف Encpsulation

يهدف هذا المبدأ إلى تجميع البيانات الخاصة بفئه ما وإخفائها حتى لا تكون ظاهرة للآخرين عبر إنشاء أنواع جديدة تسمى بالطبقات classes وللقيام بذلك فإننا نمنع بيانات الفئه(متغيراتها) من الظهور للمستخدم بشكل مباشر ونقدم له عوضاً عن ذلك توابع أو ما يعرف بـــ Methods لتقوم بالتواصل مع بيانات الفئه.

٢. مبدأ الوراثة Inheritance

ويقصد بها توريث صفات وعناصر فئه إلى صفات وعناصر فئه آخرى،حيث يصبح الكائن الإبن محتوياً على بعض صفات وخصائص الكائن الأب مما يوفر الوقت والجهد وذلك بإعادة استخدام الأشياء الموجودة بدلاً من إنشائها من جديد.

٣. مبدأ تعدد الأشكال Polymprphism

يحدث تعدد الأشكال عادة في الفئات المرتبطة ببعضها بسبب الوراثة،حيث يمكن استخدام نفس الدالة لتحقيق مهام مختلفة

مفاهيم البرمجة كائنية التوجه :

المشيدات constructors:هي دوال خاصة تسمح بتمهيد الكائنات عند قيم معينه قبل استعمالها في البرنامج

الهادمات: **Deconstructors**: وهي دوال لها وظيفة معاكسة للمشيدات وتنفذ عند الخروج من الفصيلة حيث تقوم بتحرير الذاكرة من الكائن ويسبق اسم دالة الهدم العلامة ~

سلاسل الرموز String of Characters

تمكننا السلاسل من التعامل مع النصوص مثل الكلمات أو الجمل أو الأسماء ...

السلسلة مؤلفة من تتابع من العناصر ذات النوع **Char** حيث أن كل عنصر يمثل حرف في السلسلة

فمثلاً المصفوفة (سلسلة الرموز التالية):

Char str [20];

يمكن أن تخزن سلسلة مؤلفة من ١٥ حرف كما في الشكل:

ليس من الضروري أن تملأ كل عناصر السلسلة فيمكن أن تحتوي السلسلة في نقطة ما من البرنامج الكلمة "**Hello**"التي تملأ ٥ خانات أو العبارة "**Hello World**"التي تملأ ١٢ خانة .

وعند نهاية السلسلة المخزنة في كلا الحالتين نصل إلى الرمز **Null** الذي يمكن أن يكتب ٠ أ' \0 '

أما العناصر الواقعة في المنطقة الرمادية فقيمها غير محددة

التعامل مع السلاسل

وهي تخضع لنفس قواعد المصفوفات فإذا أردنا أن نضع القيم في السلسلة أثناء التصريح نكتب :

```
char mystring[6] = { 'H', 'e', 'l', 'l', 'o', '\0' };
```

أو

```
char mystring[ ] = { 'H', 'e', 'l', 'l', 'o', '\0' };
```

ولكن هناك طريقة أخرى لعمل ذلك وهي :

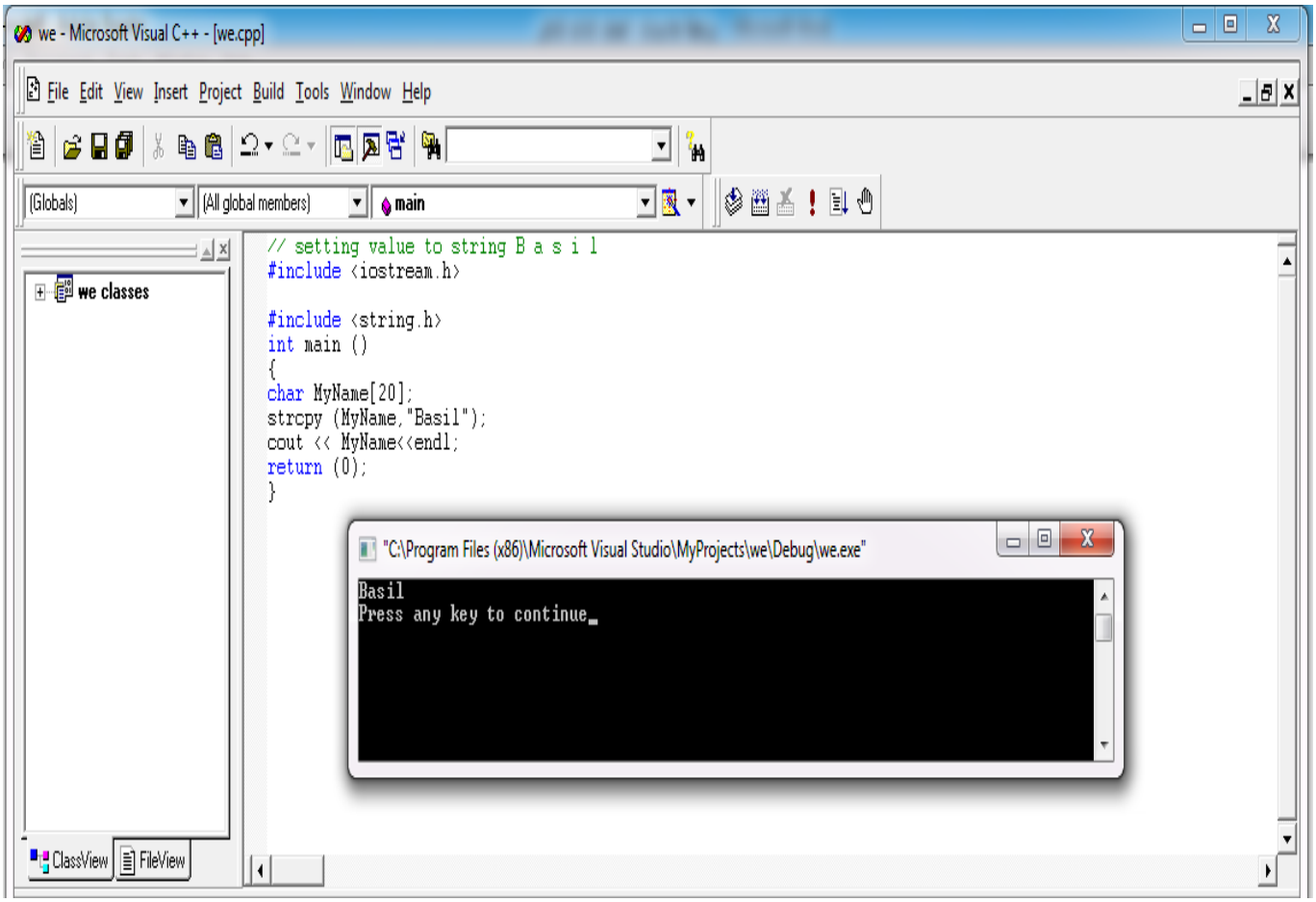
```
char mystring [ ] = "Hello";
```

حيث أن السلاسل المحصورة بعلامات اقتباس مزدوجة يضاف إلى نهايتها الثابت **null** بشكل آلي.

إن اسناد الثوابت النصية مثل"**Hello**" للسلاسل صحيح فقط أثناء الإعلان عن المصفوفة لذلك فالعبارات التالية غير صحيحة:

```
mystring = "Hello";
mystring[] = "Hello";
mystring = { 'H', 'e', 'l', 'l', 'o', '\0' };
```

مثال :



دالات السلاسل :

توجد عدة دالات تعمل على السلاسل، إذا أردنا استعمال أي من هذه الدوال في برنامج يجب أن نقوم بتضمين ملف الترويسة **string.h** ومن هذه الدالات :

١. strlen:-

تعيد الدالة **strlen( )** طول السلسلة الممررة .

٢. Strcpy:-

تستعمل الدالة **strcpy( )** لنسخ سلسلة إلى سلسلة أخرى .

٣. Strcat :-

تقوم الدالة **strcat( )** بتجميع السلاسل النصية معاً، فمثلاً إذا ألحقنا السلسلة **science** بالسلسلة **computer** ستكون نتيجة السلسلة **computer science**

٤. strcmp :-

الدالة **strcmp** تقارن السلسلة الممررة إليها كوسيطة أولى مع السلسلة الممرة إليها كوسيطة ثانية، وترجع **0** إذا كانتا متطابقتين وقيمة سالبة إذا كانت السلسلة الأولى أصغر من السلسلة الثانية وقيمة موجبة إذا كانت السلسلة الأولى أكبر من السلسلة الثانية.

التعامل مع ملفات الادخال والاخراج File I/O

يجب أن تكون هناك قناة لتمرير هذه البيانات من البرنامج إلى الملف والعكس صحيح وهي إما:

١ -إما أن تكون قناة إخراج: أى من البرنامج إلى الملف

٢ -وإما أن تكون قناة إدخال: أى من الملف إلى البرنامج

٣ -وإما أن تكون قناة إدخال وإخراج معا: أى من البرنامج إلى الملف والعكس صحيح

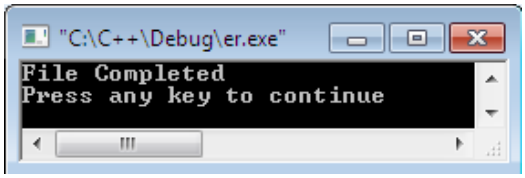
ولذلك سنستخدم مكتبة #include<fstream.h>

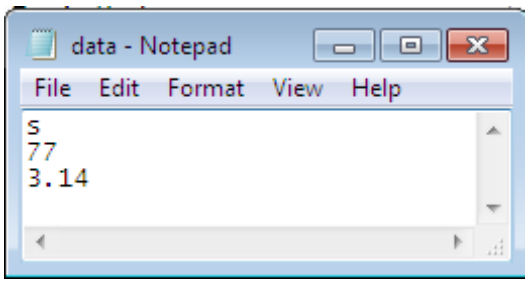
ولها ثلاث حالات	
IfStream	الملف الذي تم تعريفه هو للقراءة فقط
Ofstream	الملف الذي تم تعريفه هو للكتابة فقط
Fstream	الملف الذي تم تعريفه هو للقراءة والكتابة معاً

وتسمى القناة بأي اسم والإسم الشائع هو **Fout** لقناة الإخراج و**Fin** لقناة الإدخال

برنامج يقوم بكتابة متغيرات داخل ملف نصي اسمه data.txt يتم إنشاءه على القرص E

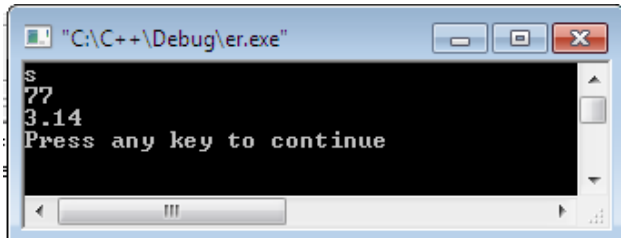
```
#include <iostream.h>
#include <fstream.h>
int main()
{
    char x = 's';
    int d = 77;
    double i = 3.14;
  // نلاحظ اننا اسندنا للمتغيرات قيم //
        ofstream fout("e:\\ data.txt");//مسار واسم الملف الذي
                                      //سيتم الكتابه فيه
    fout << x<<"\n"<<d<<"\n"<< i ;
fout.close();//لإغلاق الملف بعد الإنتهاء منه

cout << "File Completed\n";
    return 0;
}
```





برنامج يقوم بقراءة متغيرات من داخل ملف نصي اسمه data.txt موجود على القرص E

```
#include<iostream.h>
#include<fstream.h>
int main()
{
    char m;
    int i;
    double j;
    ifstream fin("e:\\data.txt");
        fin >> m >> i >> j;
        cout << m << endl
        << i << endl
        << j << endl ;
fin.close();//إغلاق الملف بعد القراءة منه

return 0;
}
```



توليد الملفات

سنستخدم مكتبة #include<stdio.h>

تعليمة فتح ملف fopen ( )

تعليمة إغلاق ملف fclose ( )

تعليمة طباعة ملف fprintf ( )

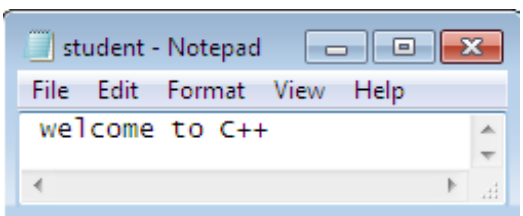
برنامج يقوم بإنشاء ملف على القرص E ويكتب داخله Welcome to C++

```
#include <stdio.h >
void main ( )
{
FILE * f;//تحديد مكان في الذاكرة
f=fopen("g:\\student.txt","w");// w معناها للكتابه داخل الملف
fprintf(f," welcome to C++");
fclose(f);
}
```

f = fopen("c:\\student.txt","w"); فللكتابة نستخدم

أما للقراءة فنستخدم "r"

وفي حال القراءة و الكتابة نستخدم "w+" و "r+"



برنامج يقوم بإنشاء ملف نصي من أجل تخزين بيانات طالب ضمن هذا الملف

```
#include<iostream.h>
#include<stdio.h>
#include<stdlib.h>
void main()
{
FILE *out;
int score;
if((out=fopen("e:\\test.txt","w"))==NULL)
{
cout<<"can not open file\n";
exit  ;
}
cout<<"enter a test score (0 terminate input)";
cin>>score;
while(score !=0)
{
fprintf(out,"%d\n",score);
cout <<"enter another score";
cin>>score;
}
fclose(out);
}
```

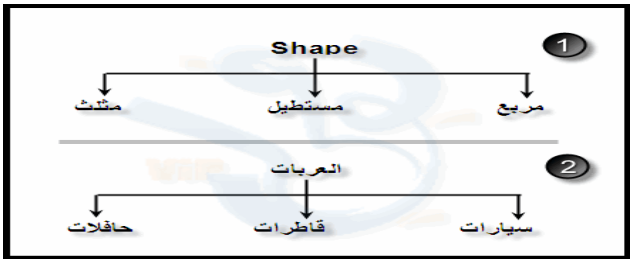
برنامج يكتب عباره داخل ملف تم إنشاءه مسبقاً

```
//نربط الملف الرأسي الذي يحوي قنوات الملفات
#include < fstream.h >
 void main()
{
   //أي للكتابة على الملف out نعرف متغير قناة خرج
   ofstream fout;
  //نفتح القناة الآن على ملف محدد باسمه
  fout.open("e:\\test.txt");
  // الآن جاهزون لاستخدام القناة
  fout << "Hello this is my first file test";
  //الآن لنغلق القناة
  fout.close();
}
```

الوراثة وتعدد الاشكال Inhertinace &Polymorphism

مفهوم الوراثة Inheritance

تعتمد فكرة الوراثة على إمكانية إنشاء فئات جديدة تكون مشتركة في صفات مع فئات موجودة أصلاً وذلك بجعل الفئة الجديدة ترث كل صفات الفئة القديمة بالإضافة إلى صفاتها الخاصة بها فبدلاً من كتابة البيانات والأعضاء الدالية المشتركة مرة أخرى في الفئة الجديدة ترث الفئة الجديدة والتي تسمى بالفئة المشتقة **Derived Class** كل البيانات والأعضاء الدالية من الفئة المُـعرفة أصلاً والتي يرمز لها بالفئة القاعدة **Base Class**.



نجد أن كل كائن تابع للفئة المشتقة هو بالضرورة تابع للفئة القاعدة ولكن العكس غير صحيح فكائنات الفئة المشتقة تحمل صفات أكثر من كائنات الفئة القاعدة ، فمثلاً فئة المستطيل هي فئة مشتقة من فئة الأشكال الرباعية وعليه يمكن القول أن أي مستطيل هو شكل رباعي ولا يمكننا القول أن أي شكل رباعي هو مستطيل.

الشكل العام لاشتقاق فئة من فئة قاعدة هو:

```
class derived-class-name : access base-class-name
{
body of class};
```

Access تحدد إمكانية الوصول إلى أعضاء الفئة القاعدة ،وهي يمكن أن تكون إما **public** أو **private** أو **protected** وإذا لم يتم تحديدها فسيفترض المترجم أن محدد الوصول هو **private** .

Class	إنشاء كلاس جديد
Drived_class	اسم الكلاس الإبن
Access	طرق الوصول إلى صنف الأساس "الأب" وتكون على إحدى حالتين: :Public نقل الصفات كما هي. :Private نقل الصفات وتحويل الصفات العامة والمورثة إلى محمية.
Base_class	اسم الكلاس الأب ، ويجب أن يكون الكلاس الأب موجوداً قبل إنشاء الكلاس الإبن.

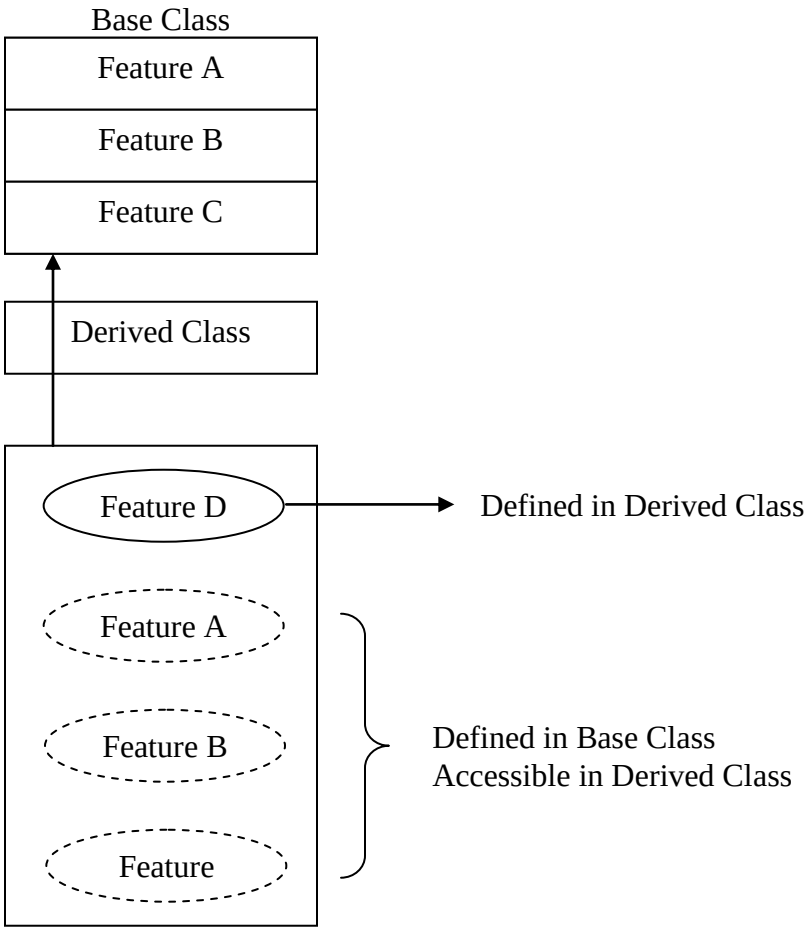
عندما يستخدم محدد الوصول **public** تسمى الوراثة عامة وفيها تتم وراثة الأعضاء العامة والمحمية في الفئة القاعدة كأعضاء عامة ومحمية في الفئة المشتقة ، عندما يستخدم المحدد **private** تسمى الوراثة خاصة ولكن في كل الأحوال الأعضاء الخاصة في الفئة القاعدة تبقى خاصة بالفئة القاعدة وعندما يستخدم محدد الوصول **protected** تسمى الوراثة محمية .

عادة تضيف الفئة المشتقة بيانات وأعضاء دالية خاصة بها وعليه تكون الفئة المشتقة أكبر من الفئة القاعدة

الوراثة المحمية Protected Inheritance

إذا كان محدد الوصول محمي **(protected)** فتسمى الوراثة محمية وعندها كل الأعضاء العامة والمحمية في الفئة القاعدة تصبح أعضاء محمية في الفئة المشتقة.

تمتلك الوراثة عدة مزايا هامة من أهمها أنها تسمح بإعادة استخدام الكود البرمجي ، إن إعادة استخدام الكود يوفر الزمن والمال وتزيد من وثوقية المبرمج لما أن الوراثة يساعد أيضاً في تحليل المشاكل البرمجية والنظرة الشمولية للتصميم العام للبرنامج كما أن أحد فوائد إعادة استخدام الكود هي تسهيل توزيع مكتبات الفصيلة **MFC** حيث باستطاعة المبرمج استخدام فصيلة مشكلة من قبل شخص آخر أو شركة أصناف جديدة مناسبة لحالته البرمجية الخاصة به . والشكل التالي يوضح مبدأ الوراثة :



- **باستخدام مبدأ الوراثة اكتب برنامج يحسب مساحة المثلث والمحيط .**

المساحة = $area = \sqrt{s(s-a)(s-b)(s-c)}$

المحيط= $a+b+c$ $s = \frac{a+b+c}{2}$

```
#include<iostream.h>
#include<math.h>
class triangle
{
protected:
     float a,b,c;
     double s,area;
public:
     void set_data(float a1,float b1,float c1)
     {
          a=a1;
          b=b1;
          c=c1;
          s=(a+b+c)/2;
          area=sqrt(s*(s-a)*(s-b)*(s-c));
     }
     void show_data()
     {
          cout<<"area="<<area<<endl;
     }
};
class cir1:public triangle
{
private:
     double circum;
public:
     void set_s()
     {
          circum=2*s;
     }
     void show_cirsum()
     {
          cout<<"circum="<<circum<<"\n";
     }
};
void main()
{
     cir1 ob1;
     ob1.set_data(10,10,10);
     ob1.show_data();
     ob1.set_s();
     ob1.show_cirsum();

}
```

- النتائج

```
area=43.3013
circum=30
Press any key to continue
```

تعدد الأشكال polymorphisme

يحدث تعدد الأشكال عادة في الفئات المرتبطة ببعضها بسبب الوراثة،حيث يمكن استخدام نفس الدالة لتحقيق مهام مختلفة هو أن دالة واحدة تأخذ أدواراً متعددة ومثال على ذلك لو أخذنا الدائرة والنقطة نجد أنها ينتميان أساساً إلى فصيلة واحدة (أن فصيلة الدائرة مشتقة من النقطة) لكنهما قد يسلكان سلوكاً مختلفاً . ولو أننا رسمنا كل منهما على الشاشة فإحداهما ترسم نقطة مفردة والأخرى ترسم نقطة متحركة حول مركز ثابت لتعطي دائرة في الشكل النهائي .

وهناك عدة طرق لتحقيق خاصية تعدد الأشكال منها استخدام التحميل الزائد للدوال **(Over Loading)** ويتم ذلك بالأسلوب الأتي :

١) أنشئ الفصيلة الأساسية (ولتكن **point**).

٢) صمم الدالة المطلوبة مع فصيلة النقطة ولتكن دالة لحساب المساحة **(surface)**.

٣) أنشأ الفصيلة المستحدثة ولتكن **(circle)**.

٤) أضف الدالة **surface** إلى الفصيلة المستحدثة (المشتقة) كعضو مع الأعضاء مع إدخال التعديلات اللازمة لكي تحسب مساحة الدائرة

مثال :

```
#include<iostream.h>
#define pi 3.14
class point
{
public :
     double surface()
     {
          return 0;
     }
};
class circle :public point
{
protected:
     float radius ;
public :
     void set_radius (float r)
     {
          radius=r;
     }
     double surface()
     {
          return pi * radius *radius;
     }
};
void main()
{
     point point1;
     circle circle1;
     float r1;
     cin>>r1;
     cout<<circle1.surface()<<endl;
     cout<<point1.surface()<<endl;
}
```

النتائج ○

```
10
3.62017e+016
0
Press any key to continue
```

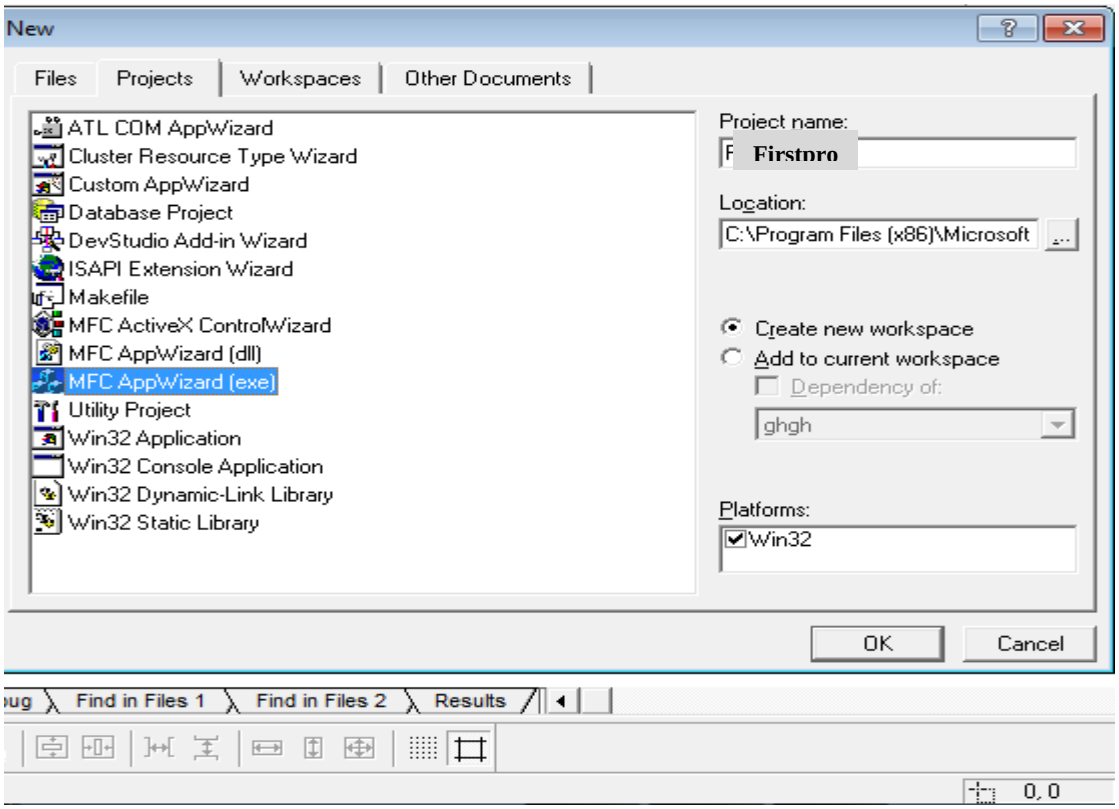
يتم إنشاء تطبيقات ذات واجهه رسوميه باستخدام معالج APP Wizard

ولإنشاء مشروع ويندوز Windows Application اتبع ما يلي:

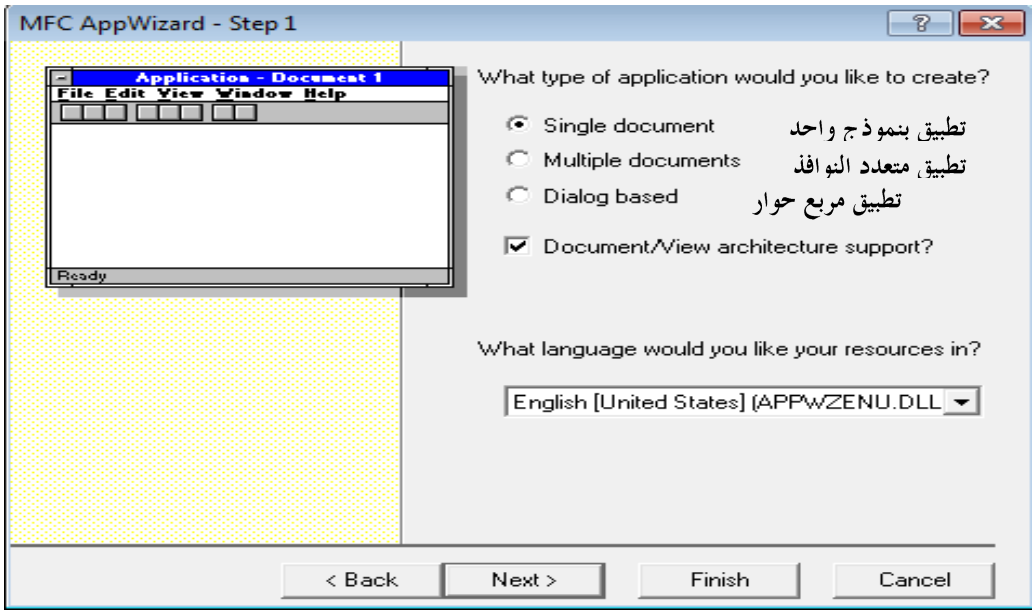
١. File –New

٢. اضغط تبويب Projects واختر MFCAppWizard.exe

٣. ادخل اسم المشروع في خانة Project Name

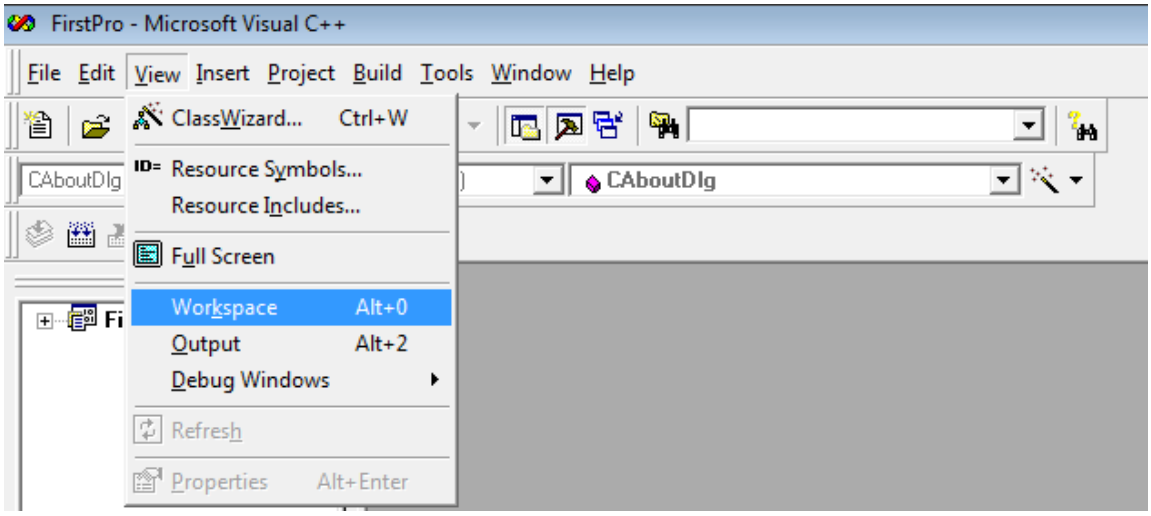


٤. اختر Single document



٥. اضغط زر Finish

٦. يظهر على يسار الشاشة مايسمى بـــ Work Space ولإظهارها أو إخفائها View- Work Space

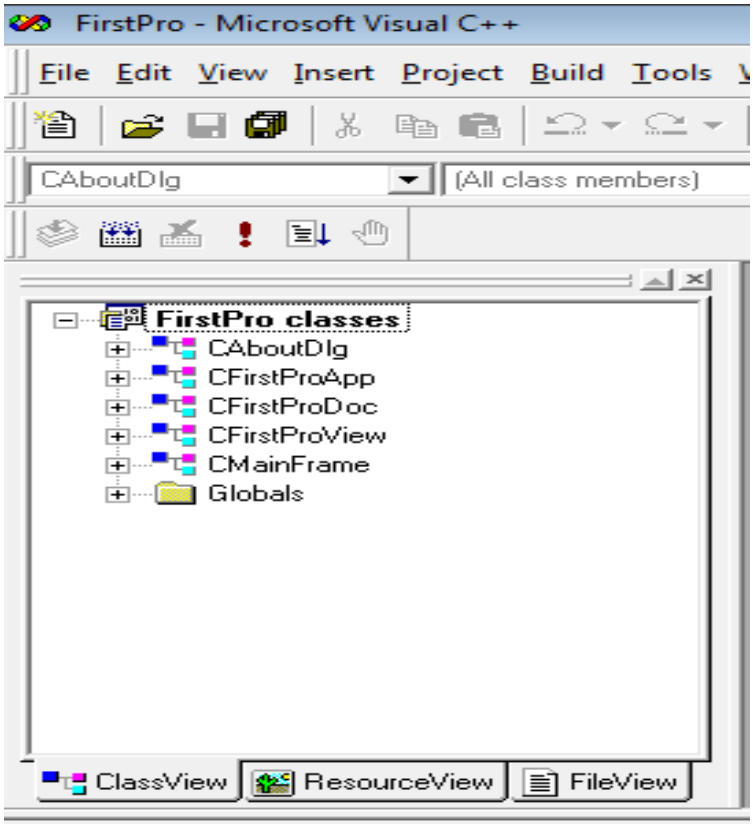


٧. تتكونانفذة Work Space من ثلاث أجزاء:

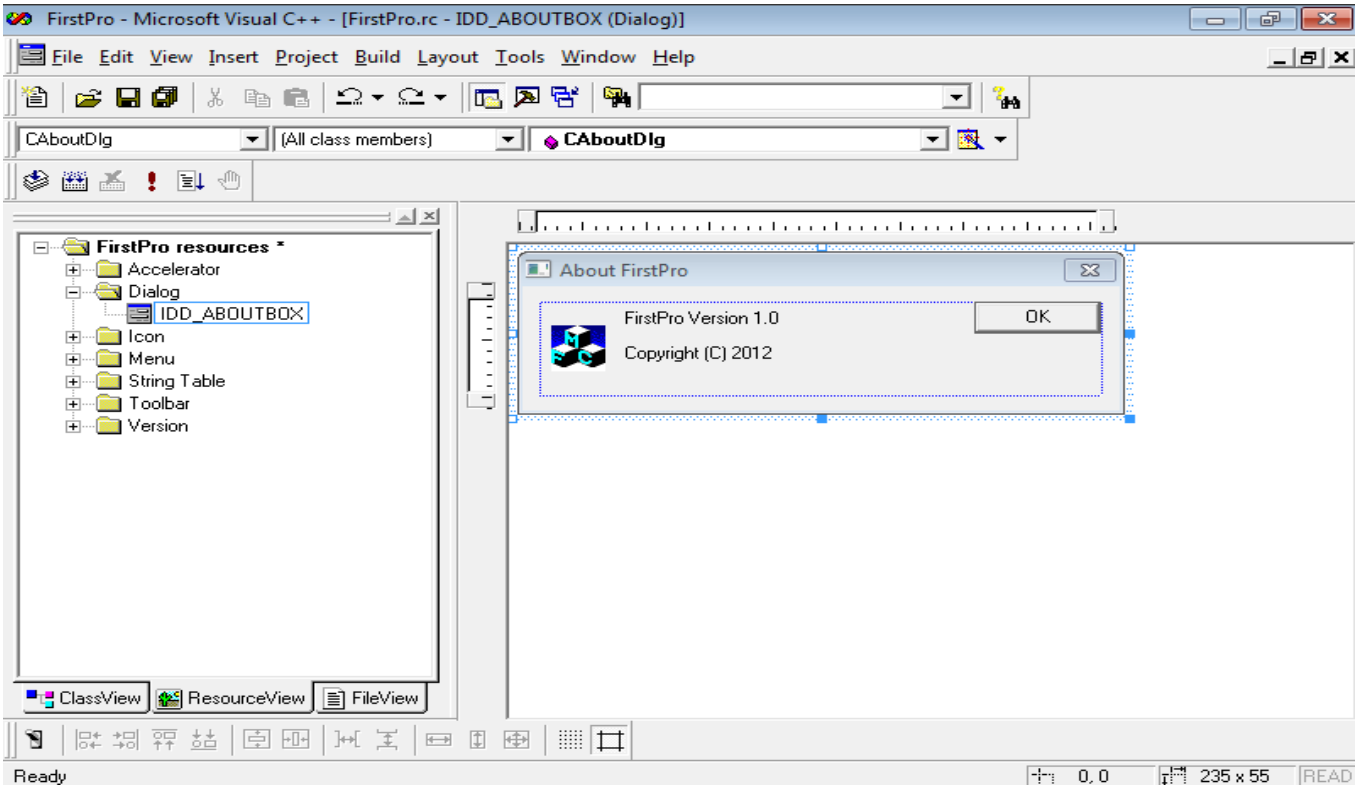
١. Class View تعرض معلومات عن الفئات المستخدمة في المشروع

٢. Resource View تعرض الموارد المتعلقة بالمشروع مثل الصور والأيقونات والنوافذ

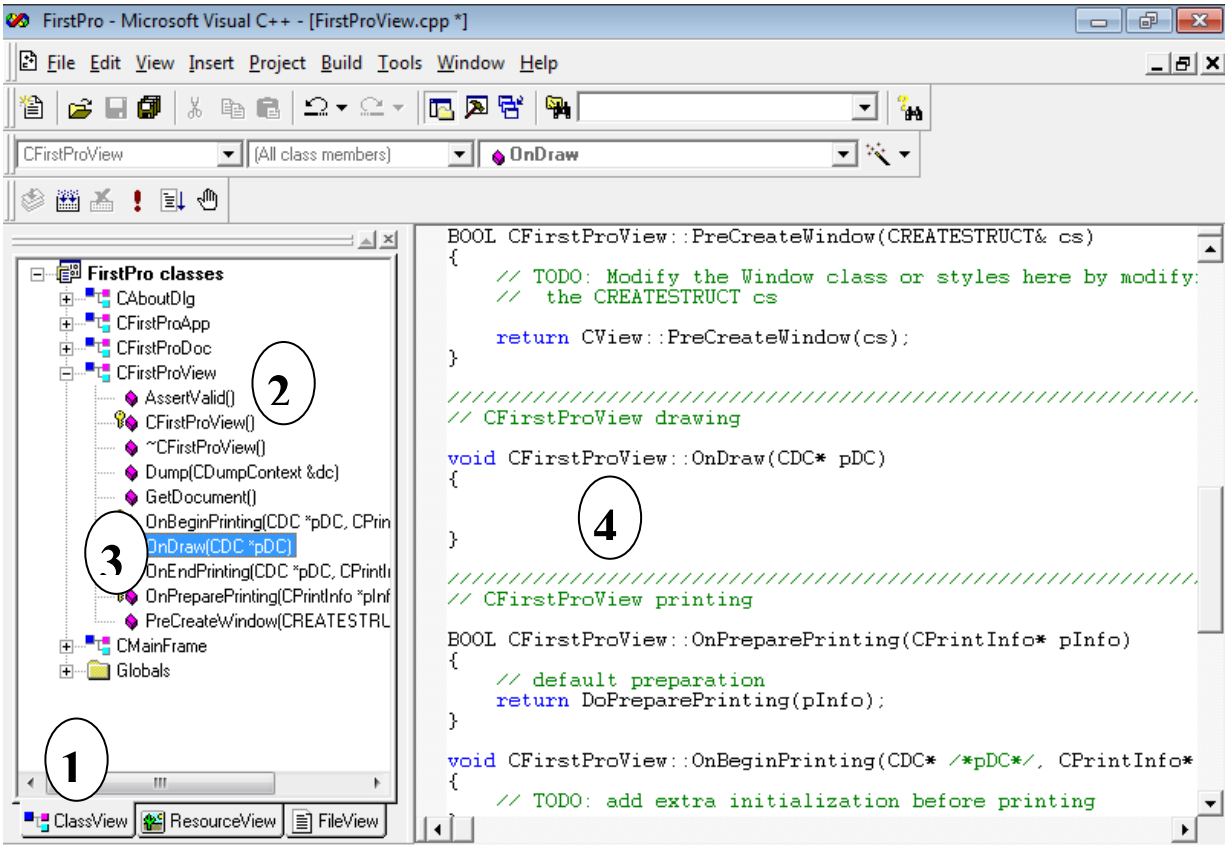
٣. File View تعرض كافة الملفات التي يستخدمها المشروع



ولفتح النافذة التي تم إنشائها إضغط على Resource View ثم مجلد Dialog



٨. ولإضافة كود رسالة تظهر عند تنفيذ البرنامجاختر Class View ثم الفئةCFirstProView



٩. امسح الأسطر الموجودة في المكان ٤ بين القوسين واكتب الكود التالي:

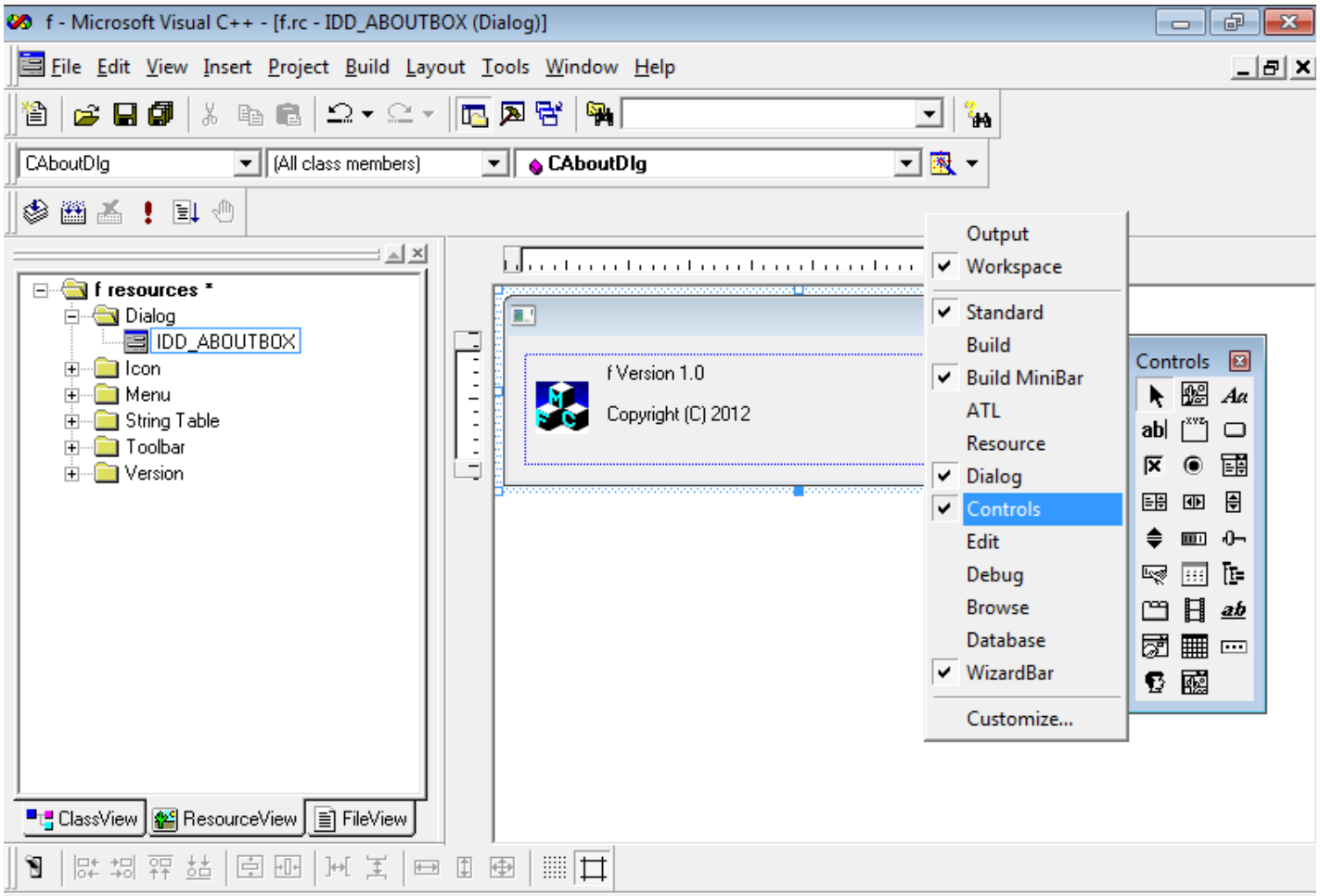
```
//////////////////////////////////////////////////////////////////
// CFirstProView drawing

void CFirstProView::OnDraw(CDC* pDC)
{
    pDC->TextOut( 50,50 ," السلام عليكم",12);
}
```

١٠. نفذ البرنامج بمفتاح F7 لعمل Compiling

١١. ولتشغيل البرنامج اضغط CTRL+F5

إضافة أدوات على النموذج (زر – مربع نص .....)

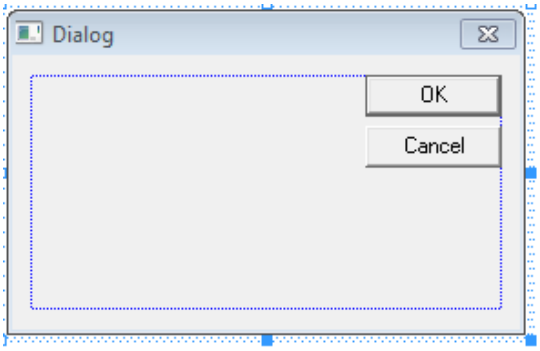
وذلك بإظهار شريط الأدوات بالضغط بالزر الأيمن للماوس في أي مكان خال على منطقة شرائط الأدوات واضافة شريط Controls



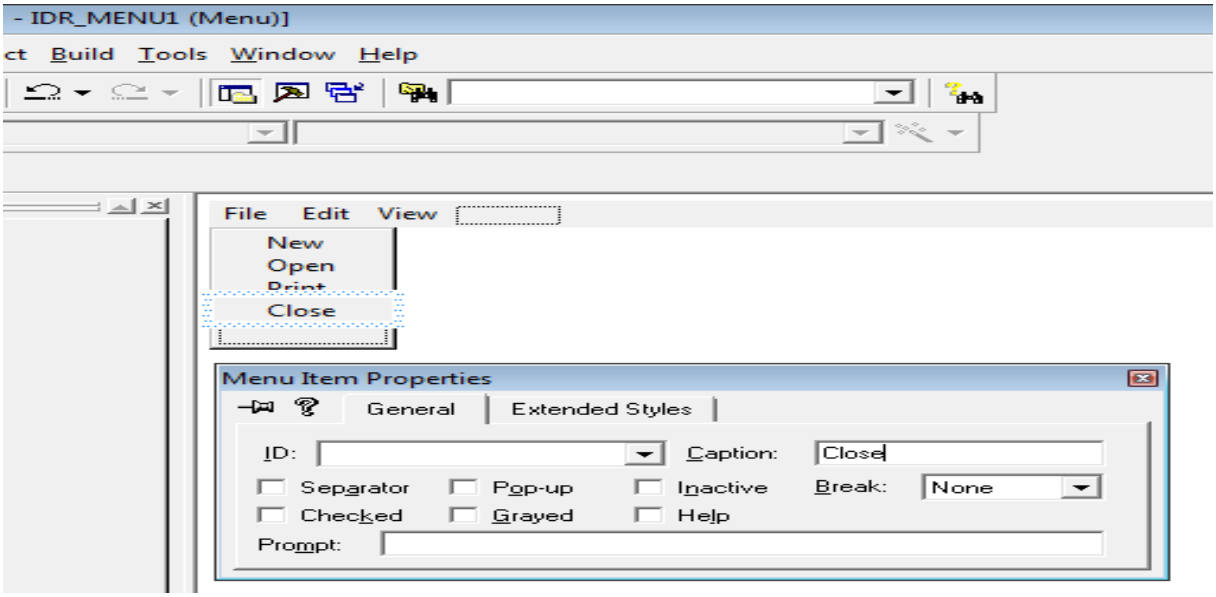
إضافة موارد إلى البرنامج

- من قائمة Insert –Resource

- لإدراج مربع حوار اضغط Dialog ثم زر New

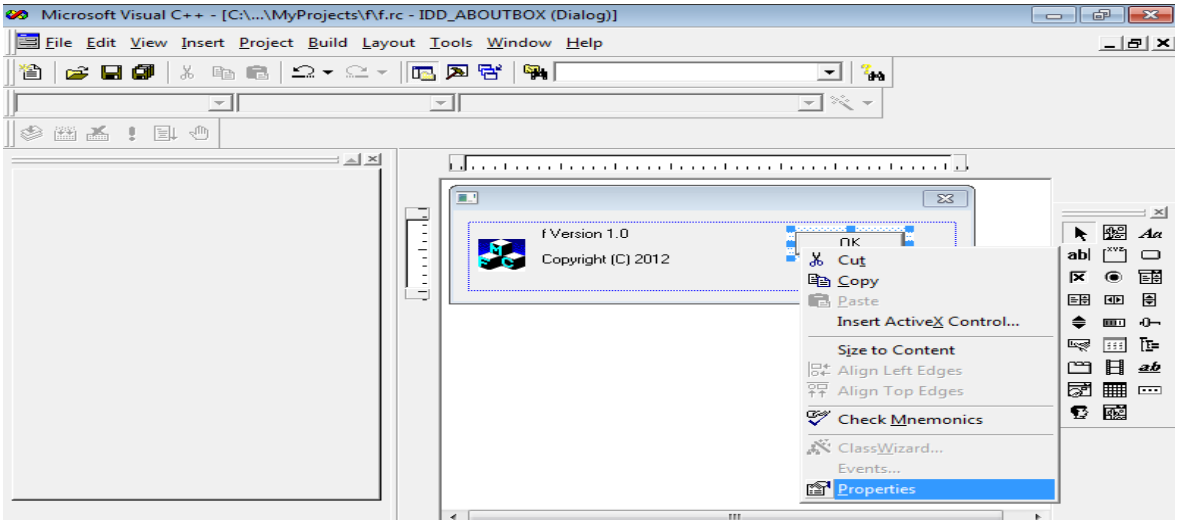


- لإدراج قائمة Menu اضغط Menu ثم زر New واكتب عناصر القائمة كما يلي



تعديل خصائص الأدوات:

اضغط بالزر الأيمن على الأداة ثم Properties



١:اكتب برنامج لطباعة الاشكال التالية:

A)

```
#include<iostream.h>
main()
{
int z,x;
for(z=1;z<=5;z++){
cout<<"*\n";
for(x=0;x<=z-1;x++)
cout<<" ";
}}
```

B)

```
#include<iostream.h>
main()
{
int z,x;
for(z=1;z<=5;z++){
for(x=z;x<=5;x++)
cout<<" ";
cout<<"*\n";
}}
```

C)

```
#include<iostream.h>
main()
{
int i,j,k;
for(i=1;i<=7;i+=2){
for(k=i;k<7;k+=2)
cout<<" ";
for(j=i;j>0;j--)
cout<<"*";
cout<<"\n";
}}
```

٢:اكتب برنامج لتقريب أي عدد تدخله

```
#include<iostream.h>
main()
{
int i,k;
float y,m;
cin>>y;
i=y;// هنا ساوينا كسر بصحيح مثلا ١.٧ يصبح ١
k=i+1;
m=i+0.5;
if(y>=m)
cout<<"near to="<<k;
else
cout<<"near to= "<<I ;
}
```

```
4.7
near to=5
```

```
5.3
near to=5
```

٣:برنامج لايجاد مضروب العدد

```
#include<stdio.h>
main()
{
int x,n,f;
scanf("%d",&n);
f=1;
if(n>=12)
printf("no factoral");
else
for(x=1;x<=n;x++)
f=f*x;
printf("factoral=%d",f);
}
```

```
13
no factoral
```

```
6
factoral=720
```

٤:برنامج لايجاد مجموع أي عدد مع الاعداد التي فوقه حتى المئة

```
#include<stdio.h>
main()
{
int x,sum,n;
sum=0;
scanf("%d",&n);
for(x=n;x<=100;x++)
sum=sum+x;
printf("The sum=%d$",sum);
}
```

```
98
he sum=297$
```

٥:برنامج جدول الضرب

```
#include<iostream.h>
void main(){
for(int i=1;i<=10;i++)
for(int j=0;j<=10;j++)
cout<<i<<"*"<<j<<"="<<i*j<<"\n";}
```

٦:اكتب برنامج لحساب عدد الارقام المدخلة اليه بحيث اذا ادخلنا ١٠٠ ينتج ٣

```
#include<iostream.h>
main()
{
int i,k,m;
cin>>k;
m=1;
for(i=1;i<=4;i++)
{m=m*10;
if(k<m)
{
Cout<<"number of bit=<<i;
break;}}
}
```

```
1989
number of bit=4
```

٧:برنامج لاخراج رواتب عمال بحيث اذا عمل ١٥٠ساعة ياخذ ثلاثة جنيهات وعلى كل ساعة عمل اضافية ياخذ جنيهان

```
#include<iostream.h>
main()
{ int i,j,k;
j=3;
cout<<"enter the hower\n";
cin>>k;
for(i=151;i<300;i++)
{ if(k<=150)
{cout<<"he cost= 3£";
break;}
j=j+2;
if(k==i)
cout<<"he cost="<<j;
}}
```

```
enter the hower
153
he cost=9$
```