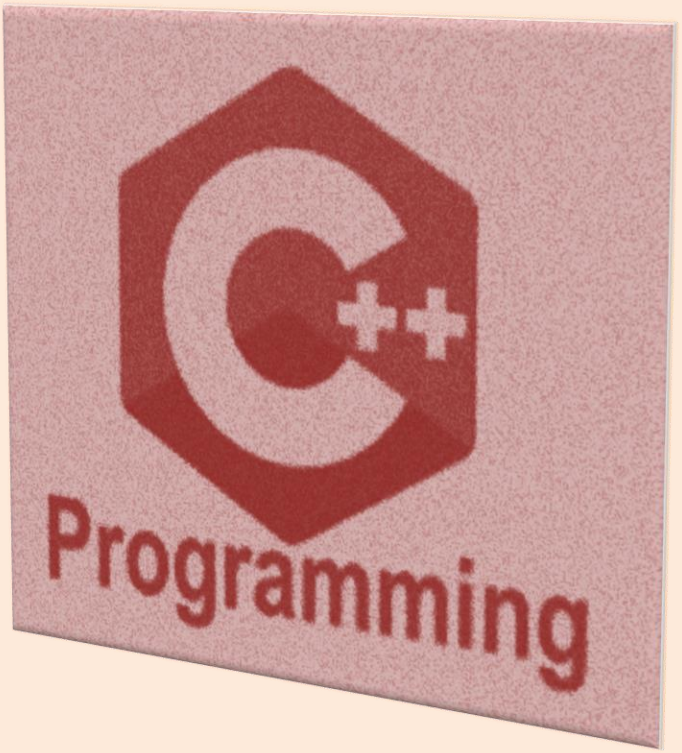
جامعة الملك سعود

مركز التدريب وخدمة المجتمع

البرمجة بلغة C++

اساسيات البرمجة والبرمجة الشيئية بلغة c++



دبلوم شبكات حاسب

سيسكو

مفردات أساسيات البرمجة

1.الخوارزميات:

- مقدمة عن مفهوم الخوارزميات .
- طرق التعبير عن الخوارزمية:

- الطريقة النصية (لغة الخوارزمية Pseudo-Code)

- الطريقة البيانية (المخططات التدفقية flowchart)

- برامج وتمارين عامة.

2.لمحة عامة حول لغة ++C:

- التعرف على لغة الآلة ولغة المجمع واللغات العالية المستوى .
- التعرف على تاريخ لغة ++C.
- التعرف على البرمجة المهيكلة Structured Programming.

3.مقدمة في البرمجة بلغة ++C:

- عمليات الاسناد.
- العمليات الحسابية وعمليات المساواة والمقارنة.
- أمثلة وتمارين بسيطة.
- بنى المعطيات البسيطة (الصحيحة – الحقيقية – المحرفية – البوليانية-...)
- التصريح عن المتحولات .
- الدخل والخرج.
- برامج وتمارين عامة.

4.نبنى التحكم Control Structures:

- بنية الاختيار if.
- بنية الاختيار if/else.
- البنى التكرارية :
  - البنية While .
  - البنية do/while .
  - البنية for.
- بنية الاختيار المتعدد switch.
- برامج وتمارين عامة .

5.المصفوفات Arrays:

- التصريح عن المصفوفات .
- أمثله عن استخدام المصفوفات .
- مصفوفات الفرز.
- المصفوفات المحرفية .
- المصفوفات المتعددة الابعاد .
- برامج وتمارين عامة .

6.السجلاتrecords

- مقدمة
- تعريف السجلات
- الوصول إلى حقول السجلات
- بناء النمط time باستخدام السجلات
- تمرير السجلات إلى التوابع
- السجلات المركبة
- مصفوفة السجلات

- إعطاء قيم بدائية للسجلات

7–الصفوف clases

- تعريف الصف
- مجل رؤية صف وكيفية الوصول إلى أعضائه
- التوابع البناءة
- التوابع البناءة بدون وسطاء
- التوابع البناءة مع وسطاء
- التحميل الزائد للتوابع البناءة
- الإسناد الافتراضي للأعضاء
- أهادمات أو المدمرات
- الدوال get وset (أو الدوال الخطية)
- الفصل بين الواجهات والنصوص البرمجية
- استخدام الوسطاء الافتراضية مع التوابع البناءة
- التوابع الأصدقاء والصفوف الأصدقاء
- توابع الوصول وتوابع الأدوات
- الأغراض الساكنة والتوابع الأعضاء الساكنة

– الأغراض الساكنة

المعطيات الأعضاء الساكنة والدوال الأعضاء– الساكن

1- الخوارزمية algorithm

أصل كلمة خوارزمية :

إن كلمة خوارزمية مشتقة من إسم العالم العربي الجليل محمد بن موسى الخوارزمي الذي عاش في بغداد من سنة 780 الى 847م في عصر الخليفة المأمون ، وقد برع هذا العالم في الرياضيات والفلك ، وترك بصـمات في التراث الحضـاري العالمي ، فقد وضـع الخوارزمي مبادئ علم الجبر وألف كتاب "الجبر و المقابلة" وأعطى الجبر اسمه حتى اصبحت كلمة الجبر موجودة في جميع اللغات تقريباً

وفي تلك الأونة انطلق اسم الخوارزميات Algorismsعلى جداول الضرب والقسمه والحساب العشري ، وظل هذا الاسم متداولاً في أوروبا مدة قرون حتى تطور مؤخراً ليحمل مدلولاً جديداً مرتبطاً بالبرمجة .

1. مقدمة :

إن أهم مرحلة في حل مسألة ما بإستخدام الحاسوب هي المرحلة المتعلقة بإيجاد خطة الحل ، يجب أن تكون هذه الخطة قابلة للتنفيذ من قبل الآلة ، وقابلة للتوصـيف على وجهٍ لا يدعو الى اللبس أو التأويل ، يطلق اسم الخوارزمية على هذه الخطة .

2. تعرف الخوارزمية:

مجموعة الخطوات المتسـلسـلة والمحدودة التي تؤدي إلى حل مسـألة معينة والوصـول إلى نتائج محددة اعتباراً من معطيات ابتدائية.

3. أنواع الخوارزميات:

1) **خوارزميات حسابية**: تهتم بالمسائل الرياضية 0 (حل معادلة من الدرجة الأولى)0

2) **خوارزميات غير حسابية**: لا تهتم بالمسائل الرياضية ولكنها تحتاج إلى حل منطقي0

(طريقة التدقيق الإملائي لنص ما،اتخاذ قراربالذهاب إلى مكان ماوتحديد الطريق الأمثل للوصول إليه)0

سنهتم في هذا الفصل بالخوارزميات الحسابية فقط0

4. طرق التعبير عن الخوارزمية :

- ***الطريقة الكلامية*** : كتابة الخوارزميات على شـكل خطوات باسـتخدام اللغة المتداولة كاللغة العربية أو الإنكليزية.

- ***<u>الطريقة الرمزية</u>*** :كتابة الخوارزميات باستخدام الرموز.
- ***<u>الطريقة التدفقية</u>*** : كتابة الخوارزميات باستخدام المخططات البيانية (المخططات التدفقية).

<u>مثال توضحي:</u>

أكتب الخوارزمية التي تعطي نتيجة حل التعبير الرياضي الآتي باستخدام اللغة المتداولة (الطريقة الكلامية):

$$Y=(x^2+7)/x(x+2)$$

علماً بأن x معلومة 0

<u>الحل:</u>

يمكن التعبير عن الخوارزمية باللغة المتداولة(العربية) على الشكل الآتي:

الخطوة الأولى :أقرأ(أدخل) قيمة المتحول x .

الخطوة الثانية: احسب المقام : $a=x(x+2)$

الخطوة الثالثة: إذا كان المقام مساوياً للصفر اطبع " المسألة ليس لها حل" 0

الخطوة الرابعة: وإلا احسب البسط : $b=(x^2+7)$

الخطوة الخامسة:احسب قيمة 0 $y : a / b = y$

الخطوة السادسة: اطبع (أكتب) قيمة y0

الخطوة السابعة: توقف 0

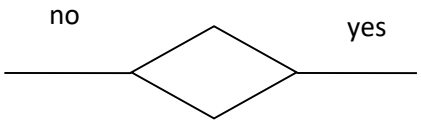
المخطط التدفقي (الهندسي أو الكابي):

لبناء المخطط التدفقي نستخدم مجموعة من الأشكال الهندسية لتسهيل هذا المخطط :

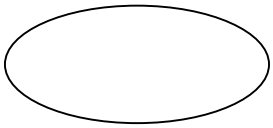
1. عمليات الإدخال والإخراج نستخدم الشكل :

2. عملية المعالجة نستخدم الشكل :

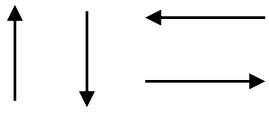
3. عملية الشرط(القرار) نستخدم الشكل :



4. لبداية ونهاية الخوارزمية نستخدم الشكل :



5. لمعرفة اتجاه الخوارزمية نستخدم الشكل :



6. نقطة توصيل وربط :

تمرين1: اكتب الخوارزمية الكلامية والرمزية والمخطط التدفقي لإيجاد مساحة ومحيط المستطيل ؟.

الحل :

الخوارزمية الكلامية :

1- المدخلات : الطول والعرض .

2 - المعالجة : المساحة(s) = الطول x العرض

المحيط (m) (الطول+ العرض) x 2

3- المخرجات : المساحة والمحيط

الخوارزمية الرمزية :

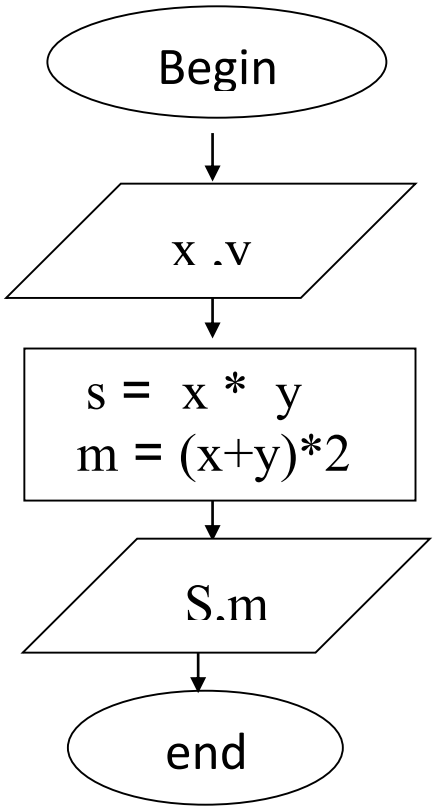
1- المدخلات : y, x

2- المعالجة : s= y*x

m = (x+y) * 2

3- المخرجات : m , s

المخطط التدفقي :



***تمرين2*: على نمط المثال السابق اكتب الخوارزمية الكلامية و الرمزية والمخطط التدفقي لإيجاد مساحة ومحيط الدائرة ؟**

الحل :

الخوارزمية الكلامية :

1. المدخلات: نصف القطر (r)
2. المعالجة : المساحة(s) = π x نصف القطر للتربيع
   s=

   المحيط (m) =2 x نصف القطر p x
3. المخرجات: المساحة والمحيط لدائرة

الخوارزمية الرمزية :

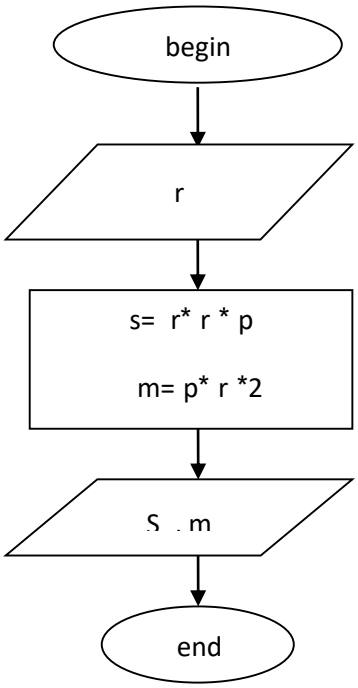
1- المدخلات : r

2- المعالجة : p* r * r

m= p* r *2

3- المخرجات : s,m

المخطط التدفقي :



تمرين3: اكتب الخوارزمية الرمزية والمخطط التدفقي لإدخال x (عدد) وإيجاد قيمة y=(x-2)/x

الحل:

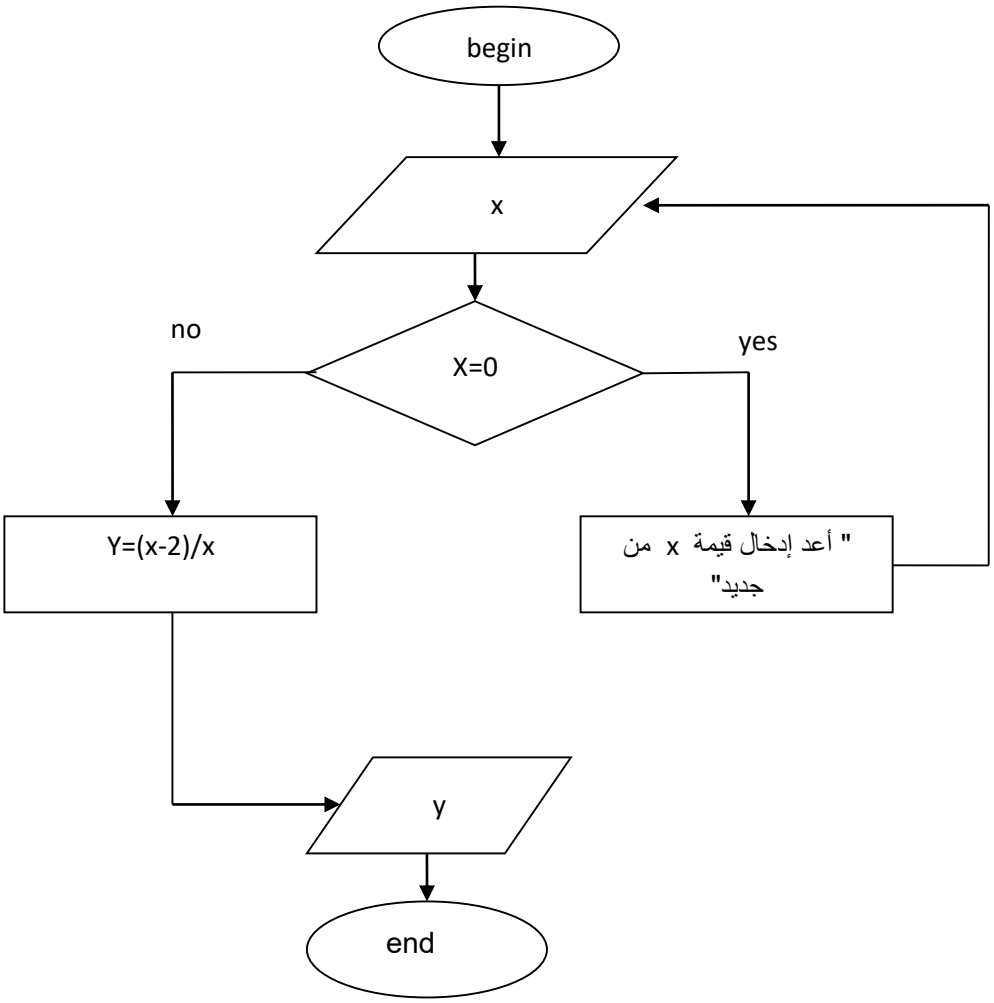
الخوارزمية الرمزية : 1) المدخلات : x

2) المعالجة : إذا كانت x=0 عندئذ " أعد ادخال قيمة x من جديد لانه لايمكن القسمة على صفر"

والا فاحسب : y= (x-2)/x

3) المخرجات : y

المخطط التدفقي :



تمرين4: اكتب الخوارزمية الرمزية والمخطط التدفقي لايجاد y=x/(x-3)

الحل:

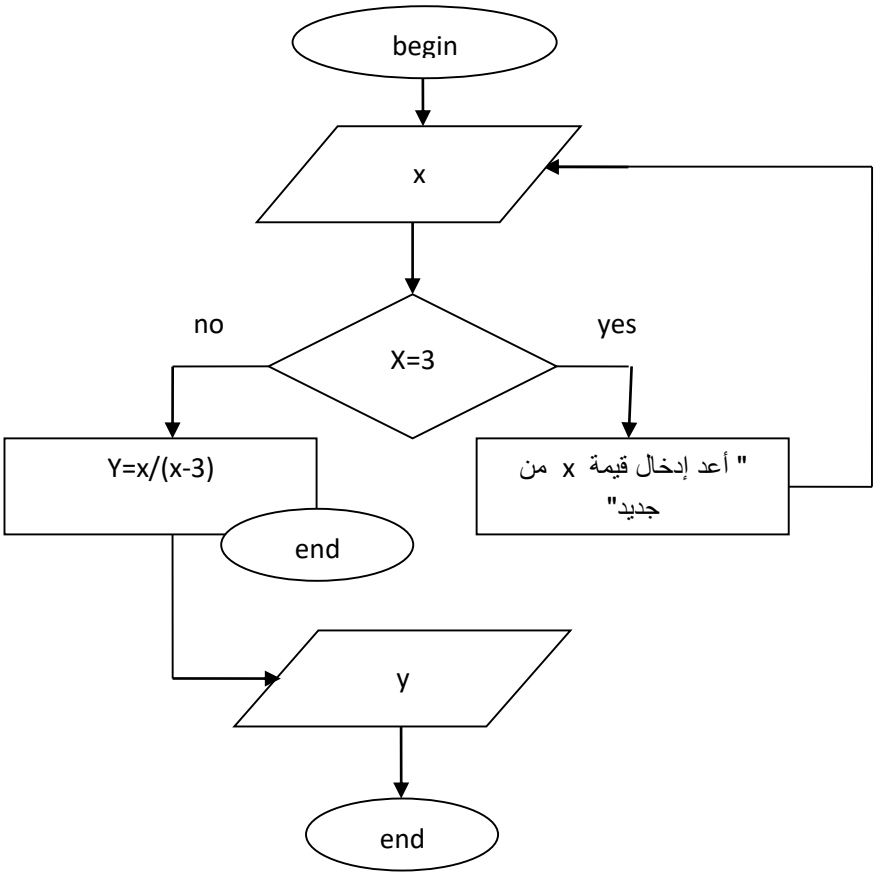
الخوارزمية الرمزية : 1) المدخلات : x

2) المعالجة : إذا كانت x=3 عند ئذ " اعد ادخال قيمة x "

والا اطبع y=x/(x-3)

3) المخرجات : y

المخطط التدفقي :



تمرين 5: اكتب الخوارزمية الرمزية والمخطط التدفقي لحل المعادلة :a X + b = 0

مناقشا جميع الحالات الممكنة لـ a,b

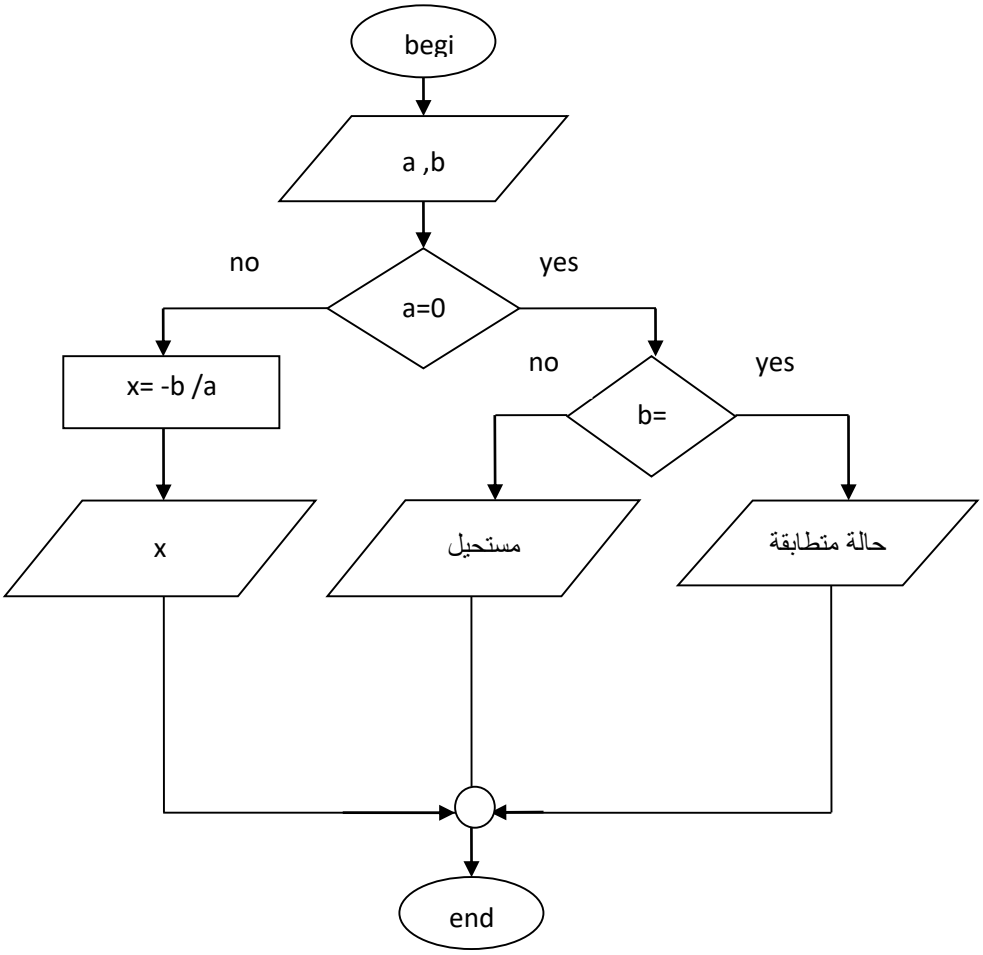
الحل :

الخوارزمية الرمزية :

- أدخل(اقرأ): a ، b .
- إذا كان (a=0 , b< >0) نجد : 0x+b=0 أي b=0
أطبع (أكتب) : "مستحيل الحل"0
- إذا كان (a=0 , b=0) نجد : 0x+0=0
أطبع (أكتب) : " حالة متطابقة "
- إذا كان (a< >0) نجد : x=-b/a

أطبع قيمة x

المخطط التدفقي :



تمرين 6: اكتب الخوارزمية الرمزية والمخطط التدفقي (الانسيابي) لإيجاد قيمة y المعطاة بالشكل التالي :

$$Y = \begin{cases} 2/(x-2) & x>2 \\ -4/(5-x) & x<=-2 \end{cases}$$

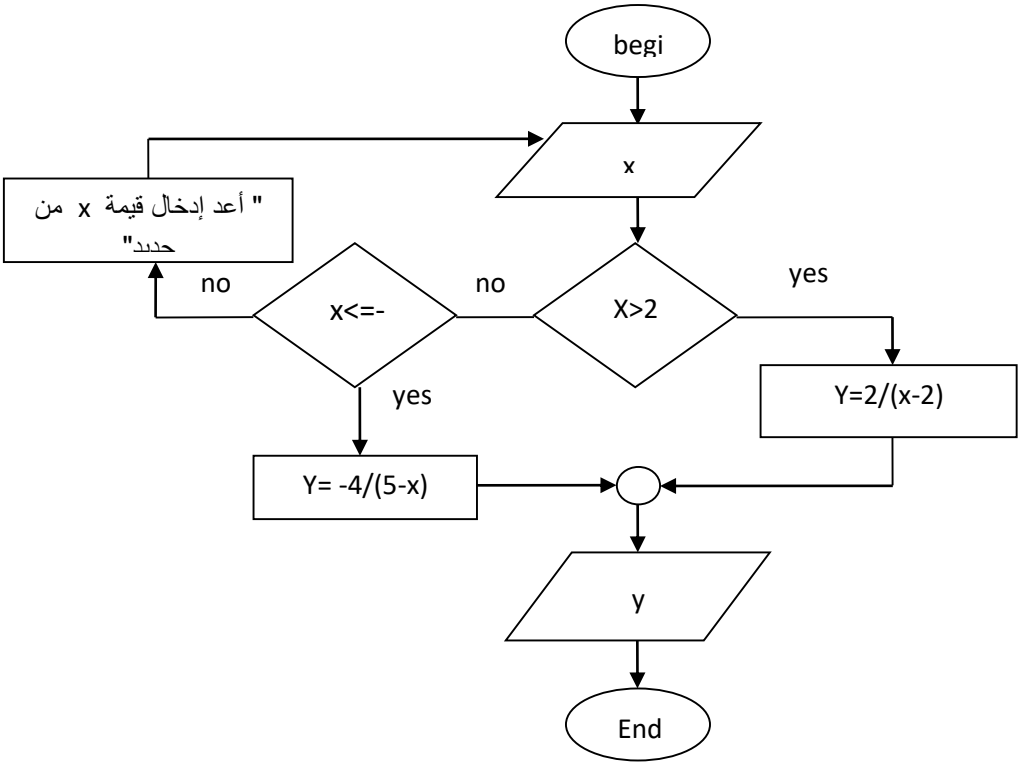
الحل :

الخوارزمية الرمزية :

1- المدخلات : x

2- المعالجة : إذا كانت x>2 عندئذ y=2/(x-2)
وإلا إذا كانت x<=-2 عندئذ y=-4/ (5-x)
وإلا " اعد ادخال قيمة x "
3- المخرجات : y

المخطط الانسيابي (التدفقي ، الصندوقي) :



تمرين 7: اكتب الخوارزمية والرمزية والمخطط التدفقي لحل معادلة الدرجة الثانية

$$aX^2+bX+c=0$$

الحل :

الخوارزمية الرمزية :

1) أدخل (اقرأ) : a,b,c
2) إذا كان (a=0) نفذ :

تصبح المعادلة معادلة من الدرجة الأولى : bX+c=0

i) إذا كان (b = 0) نفذ :

C =0 حالة متطابقة

C < >0 حالة مستحيلة

ii) إذا كان (b <> 0) نفذ :

X= -c/b

3) إذا كان (a<> 0) نفذ :

حساب D دالتا : $D=b^2 -4*a*c$

i) إذا كان (D = 0) نفذ :

أطبع : " للمعادلة جذران متماثلان "

وأحسب:- $X_1 = X_2$=
b/2*a

ii) إذا كان (D < 0) نفذ :

أطبع : " للمعادلة جذران عقديان "

iii) إذا كان (D > 0) نفذ :

أطبع : " للمعادلة جذران حقيقيان "

وأحسب: X_1=(-b-sqrt(D))/(2*a)
X_2=(-b+sqrt(D))/(2*a)

<u>المخطط التدفقي:</u>

تمرين 8: اكتب الخوارزمية الرمزية والمخطط التدفقي لإدخال عدد صحيح (x) موجب وطباعة إذا كان فرديا أم زوجيا ؟

الحل :

الخوارزمية الرمزية :

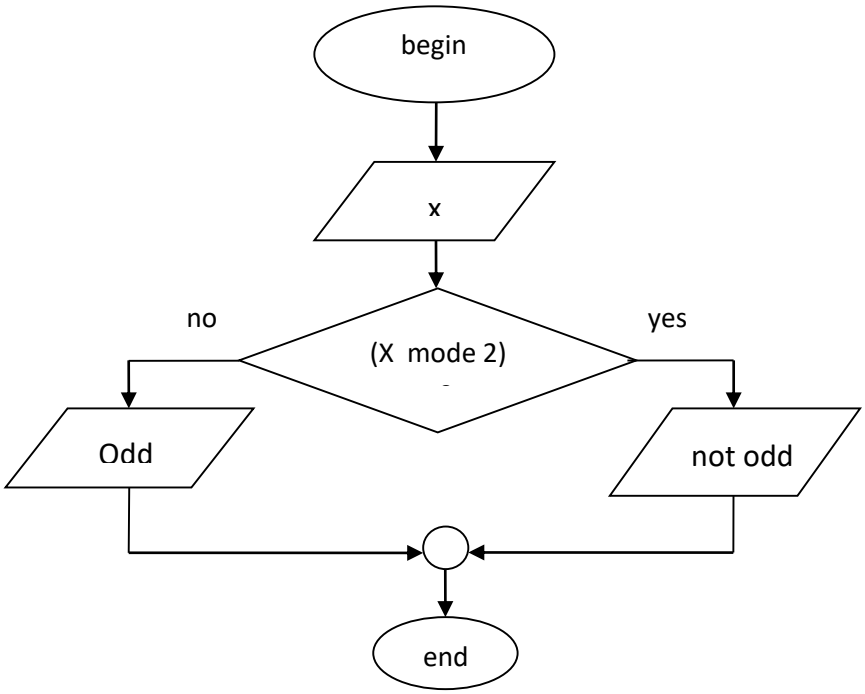
1. المدخلات: x
2. المعالجة والمخرجات: إذا كان باقي قسمة العدد على 2 يساوي صفر (x mode 2= 0) فإن :

أطبع " العدد زوجياً أو not odd "

وإلا فإن :

أطبع " العدد فردياً أو odd "

المخطط التدفقي :



تمرين 9: اكتب الخوارزمية الرمزية والمخطط التدفقي لإدخال عشرة أعداد مختلفة وإيجاد المتوسط والمجموع ؟

الحل :

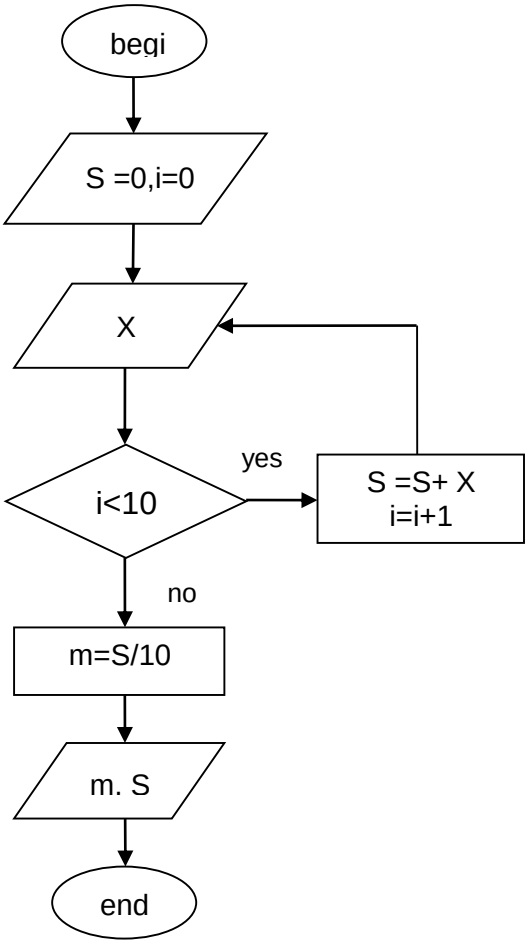
الخوارزمية الرمزية :

1. المدخلات : x ؛ S =0 ؛ i=0
2. المعالجة : العداد (i=i+1); المجموع (S=S+x)
إذا كان i<10 عندئذ " أعد إدخال
" وإلا i >=10 عندئذ " توقف عن الإدخال i "

m=S/10

3. المخرجات : المجموع (s) ، المتوسط (m)

المخطط التدفقي :



تمرين10: اكتب الخوارزمية الرمزية والمخطط التدفقي لإدخال عشرة أعداد وطباعة الفردي منها فقط ؟

الحل :

الخوارزمية الرمزية :

1- المدخلات : X, i=0

2- المعالجة و المخرجات: إذا كان i<10 عندئذ

وإذا كان X mode 2=0 عندئذ

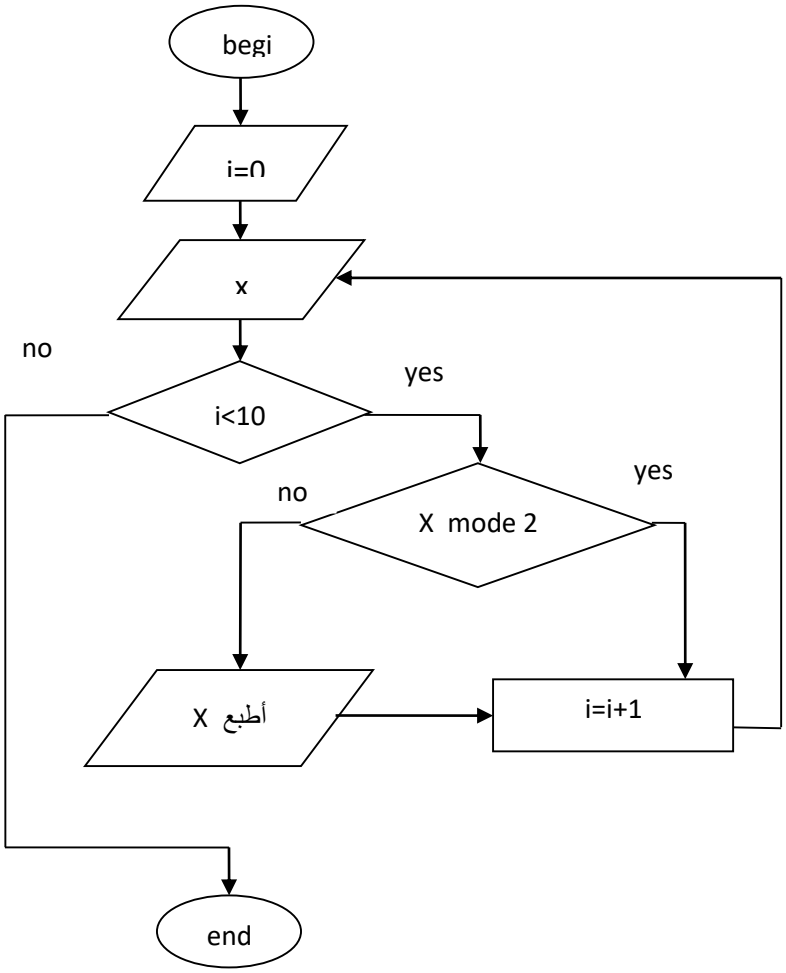
i=i+1 و" أعد إدخال x "

وإلا أطبع() أكتب قيمة x الحالية ثم أدخل قيمة جديدة لـ x

i=i+1 وشغل العداد0

وإلا أخرج من البرنامج

المخطط التدفقي :



مقدمة في البرمجة بلغة C++

* أنواع اللغات :

يمكن تقسيم اللغات المستخدمة في البرمجة إلى ثلاثة أنواع:

1- لغة الآلة ، 2. لغة المجمع 3. اللغات العالية المستوى

1. لغة الآلة:

هي اللغة التي يستطيع الحاسب أن يفهمها مباشرة وهي معرفة من قبل البنية الصلبة للحاسب ، تتألف بشكل عام من سلاسل من الأعداد (مجموعات من الأصفار والواحدات) التي تعطي الأوامر للحاسب من أجل تنفيذ تعليماته الأولية كل تعليمة على حده.

ترتبط هذه اللغة ارتباطاً وثيقاً بالآلة machine-dependent وهذا يعني أن لغة آلة ما لا تستخدم إلا لنفس النوع من الآلات فقط .

2. لغة المجتمع :

هي لغة تستخدم مصطلحات قريبة من اللغة الإنكليزية للتعبير عن العمليات الأولية للحاسب، وقد تم تطوير مترجمات للبرامج تسمى بالمجمعات assemblers تحويل البرامج من لغة المجمع إلى لغة الآلة.

3. اللغات العالية المستوى :

هي اللغات التي ظهرت لتسريع عملية البرمجة وذلك باستخدام تعليمات تقوم بالعديد من المهام الجوهرية ، وتهد اللغات C++,c من أكثر اللغات العالية المستوى قوة وانتشاراً .

– تدعى البرامج التي تقوم بتحويل النصـــوص من البرامج مكتوبة بلغات عالية المســـتوى إلى لغة الآلية بالمترجمات.

ملاحظات:

1- ÷ يوجد بعض المفسرات interpreter programs التي تستطيع تنفيذ البرامج المكتوبة بلغات عالية المستوى مباشرة دون الحاجة إلى ترجمة هذه البرامج إلى لغة الآلة.

2- البرامج المترجمة هي أسرع تنفيذاً من البرامج المفسرة عموماً .

* البرمجة بلغة C++:

تسهل لغة C++الأسلوب المهيكل والمنهجي لعملية تصميم البرامج ، حيث تتألف برامج هذه اللغة من مكونات تسمى الصفوف classesوالتوابع Functions وبالتالي يمكن تقسيم عملية تعلم لغة الـــــC++ إلى قسمين : يعتمد الأول منها على تعلم لغة الـــــC++ نفسها في حين يسمح الثاني بتعليم كيفية استخدام الصفوف الملحقة بهذه اللغة واستخدام التوابع الموجودة ضمن المكتبة المعيارية ANSI C.

*** مراحل تنفيذ برامج C++ :**

يتم التنفيذ خلال ست مراحل هي بالشكل التالي:

- **مرحلة الكتابة ضمن Edit :** وهي كتابة نص البرامج في أي محرر نصوص يستخدم لكتابة البرامج بلغة C++ .
- مرحلة ما قبل الترجمة ؛ Preprocess : هي تصحيح البرنامج من الأخطاء ومن ثم تخزينه على وحدة تخزين ثانوية مثل الأقراص بتوسع CPP, CXX وذلك حسب بيئة العمل.
- **مرحلة الترجمة Compile :** هي ترجمة البرنامج إلى لغة الآلة.
- **مرحلة الوصل Linking:** تتضمن برامج الـــــ C++استدعاءات لتوابع تم تعريفها في مكان آخر مثل المكتبات المعيارية ، وبالتالي مهمة هذه المرحلة هي استخدام الواصل Linker لوصل الملف مع نصوص التوابع الناقصة من أجل الوصول إلى صورة قابلة للتنفيذ .
- **مرحلة الشحن Loading :** قبل تنفيذ البرنامج يجب وضعه في الذاكرة وذلك باستخدام الشاحن Loader الذي يقوم بأخذ الملف التنفيذي ونقله إلى الذاكرة.
- **مرحلة التنفيذ Execute:** هي مرحلة التنفيذ التي تتم تحت إشراف وسيطرة وحدة التحكم والمعالجة CPU .

* أمثلة بسيطة : لتعلم مبادئ أساسية في لغة C ++ –

1– طباعة نص مؤلف من سطر :

كل الكتابات التي تلي هذه الإشارة (//) تسمى تعليق لا يتم تنفيذه // First Program

توجيه ما قبل الترجمة حيث يتم ضمن محتوى (.h) // # include<iostream.h>

الملف الرأسي ذو الامتداد الحاوي على العمليات الخاصة بالدخل والخرج لنص البرنامج

التابع الرئيسي الذي يبدأ من عند التنفيذ // main ( )

بداية البرنامج // {

تعليمة الطباعة // cout << " welcome to c++ " ;

```
    return 0 ;                                          إحدى طرق الخروج من التابع //
    }                                                   نهاية البرنامج //
```

```
(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
welcome to c++
```

2 . برنامج جمع عددين صحيحين :

```
# include < iostream.h>
main ( )
{
  int x1 , x2, x3 ;                  // تعريف المتحولات
  cout <<" enter first numbe ";      // تعليمة الطباعة
  cin >> x1 ;                        // تعليمة قراءة متحول
  cout << " enter second number ";
  cin >> x2
  x3 = x1 + x2 ;                     //x3 إجراء عملية الجمع والإسناد إلى المتحول الجديد
  cout << "sum is " <<x3 ;           // تعليمة الطباعة المتعددة
return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
enter first numbe 10
enter second number 55
```

sum is **65**

ملاحظة:

1- مفهوم يتعلق بالذاكرة ألا وهو طريقة حجز المتحولات:

كل اسم من أسماء المتحولات مثل .... , x3 ,x1, x2 يتم وضعه في الذاكرة ويعرف بإسم name ونمط type وحجم size وقيمة value وبالتالي فإن المتحول x1 يملك الاسم x1 يملك الاسم x1 والنمط int والحجم 2 بايت والقيمة هي حسب القيمة المقروءة .

x1	5
x2	10
x3	15

مواضع المتحولات في الذاكرة مع ذكر الاسم والقيمة

2. أنواع المتحولات:

- المتحول التعدادي enum
- المتحول المحرفي char
- المتحولات الصحيحة short int, int , long int , unsigned sort int, unsigned int , unsigned long int.
- المتحولات الحقيقة float , double , long double

وبين الجدول التالي أنواع المتحولات ومجالاتها :

نوع المتحول	المجال
char	-128 to 127

int	−32768 to 32767
unsigned int	0 to 65535
short int	−32768 to 32767
Unsigned short int	0 to 65535
Long int	−2147483648 to 214748364
float	−3.4E−38 to 3.45E+38
double	−1.7E−308 to 1.7E+308
long double	−3.4E−4932 to 1.1E+4932

* **العمليات الحسابية** :

اسم العملية	الرمز الحسابي	طريقة التعبير حسب لغة C++
الجمع	+	x1 + x2
الطرح	−	x2 − x1
الضرب	*	x1 * x2
القسمة	/	x1 / x2
باقي القسمة الصحيحة	%	x1 % x2

تقوم C++ بتطبيق العمليات في العبارات الحسابية حسب ترتيب معين محدد تبعاً لقواعد الأولوية بين العمليات التي تماثل قواعد الأولوية في الجبر وذلك كما في الجدول التالي:

العملية	اسم العملية	ترتيب عملية التقسيم (الأولوية)
( )	الأقواس	**تقييم أولاً ، إذا وجد في العبارات الحسابية أقواس متداخلة ضمن بعضها البعض فالحساب يبدأ انطلاقاً من أول مجموعة في الداخل أما إذا كان لدينا مجموعة من الأقواس جانب بعضها البعض وعلى نفس المستوى عندها يبدأ الحساب من اليسار إلى اليمين.**
* ، / ، %	الضرب، القسمة ، باقي القسمة	**تقييم ثانياً ، إذا وجدت على نفس المستوى فإنها تقييم من اليسار إلى اليمين .**
+ ، −	**الجمع ، الطرح**	**تقييم في النهاية ، إذا وجدت على نفس المستوى فإنها تقييم من اليسار إلى اليمين .**

أما بالنسبة لعمليتي الإسناد والمقارنة فتتم بالشكل التالي: جميع العمليات الحسابية يتم تجميعها من اليسار إلى اليمين إلا عملية الإسناد تتم من اليمين إلى اليسار .

الشكل الجبري	الشكل الموافق حسب c++	مثال	معنى الكتابة
=	= =	x = = y	x تساوي y
≠	! =	x != y	x لا تساوي y
<	<	x <y	x أصغر من y
>	>	x >y	x أكبر من y
≤	<=	x < = y	أصغر أو يساوي y
≥	>=	x > = y	x أكبر أو يساوي y

***العملية المنطقية Logical operators :**

وهي ثلاثة :

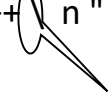
And يرمز لها &&

Or يرمز لها ||

Not يرمز لها !

*** سلاسل الهروب :**

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



حرف الهروب

```
return 0;
}
```

يدعى \ بحرف الهروب وهو يلحق بحرف يدل على معنى معين كما هو موضح في الجدول:

سلسلة الهروب	المعنى
\ n	سطر جديد أي وضع المؤشر في بداية السطر التالي
\ t	تحريك المؤشر مسافة جدولية أفقية
\"	تستخدم لطباعة علامة الاقتباس

*** بعض الأمثلة:**

1- أكتب برنامجاً يأخذ كدخل ثلاث أعداد صـحيحة من لوحة المفاتيح ثم يطبع مجموعها ومتوسـطها وناتج جداؤها.

```
# include < iostream.h>
main ( )
{
int a , b, c ;
cout << " enter a =" ; cin >> a ;
cout << " enter b = " ; cin >> b ;
cout << " enter c = " ; cin >> c ;
cout << " sun is " << a+b+c << " \ n" ;
cout << average is " << ( a+b+c)/3 <<" \n";
cout << product is " << a * b* c;
return ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME02.EXE)
enter   a = 10
enter   b = 20
enter   c = 33
sun is          63
average is      21
product is      6600
```

2- أكتب برنامج يقرأ نصف قطر دائرة ثم يطبع قيمة قطر الدائرة ، محيطها ، مساحتها .

ملاحظة : قيمة π =3.14

```
# include <iostream.h>
main  ( )
{
float  r ;                    //  تعريف متحول حقيقي
float p = 3 , 14 ;            // تعريف متحول حقيقي وإسناد قيمة له
cout << " enter r =" ; cin >> r ;
cout << r * 2=" << r * 2<<"\n";
cout <<"2*p*r = " << 2*p*r<<"\n" ;
cout << "p*r*r =" << p*r*r;
return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME02.EXE)
enter  r = 4.5
r * 2 = 9
2 * p * r = 28.26
p*r*r = 63.585
```

3- أكتب برنامجاً يقوم بطباعة مستطيل

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

```
{
  cout << " ***********\n" ;
  cout << " *\t " <<" *\n";
  cout << " *\t " <<" *\n";
  cout << " *\t " <<" *\n";
  cout << "***********\n";
return 0;
}
```

```
  (Inactive C:\TCWIN45\BIN\NONAME04.EXE)
***********
*      *
*      *
*      *
***********
```

3.بنى التحكم Control Structures:

1.البنى الشرطية:

* بنية الأختيار if :

تقوم بنية الاختيار ifبتنفيذ فعل معين عندما يكون الشرط المرافق لها محققاً وإلا يتم تجاهله ، ولها الشكل العام التالي :

if (condition) statement :

مثال 1:

علامة النجاح في أحد الامتحانات تساوي 60 درجة عندها فإن تعليمة الـ if تكون بالشكل:

```
if  ( grad > = 60 ) cout <<"passed";
```

مثال 2:

أكتب برنامجاً يطلب من المستخدم إدخال عددين صحيحين .ثم يأخذ العددين ليطبع العدد الأكبر بينهما متبوعاً بالرسالة "is larger" إذا كان العددان متساويين عندها يطبع البرنامج الرسالة "the number are equal"

```
#include <iostream.h>
main ()
{
    Int a, b;
    cout<<"enter" a=";cin>>a;
    cout<<"enter" b=";cin>>b;
    if ( a > b ) cout <<a<< " is larger" ;
    if ( a < b ) cout <<b<<" is larger" ;
```

```
    if ( a = = b ) cout <<"the numbers are eonal"
    return 0 ;
}
```

```
Inactive C:TCWIN45\BIN\NONAME05.EXE)
enter  a = 100
enter b = 69
100 is larger .
```

* بنية الأختيار if/else

تسمح بنية الاختيار if / else بتحديد جملة من الأفعال الممكن تنفيذها إذا كان الشرط المرافق صحيحاً أو إذا لم يكن كذلك ، ولها الشكل العام التالي:

```
if ( condition )
    statement 1 ;
else
    statement 2;
```

مثال 1:

إذا كان علامة الطالب أكبر أو يســاوي القيمة 60 درجة فيطبع كلمة "passed" وإلا فهي تطبع الكلمة "failed" عندها فإن تعليمة الـ if / else تكون بالشكل

```
if ( grad > = 60 )
    cout << " passed " ;
else
    cout << "failed" ;
```

مثال 2:

أكتب برنامجاً يقرأ عدداً صحيحاً ثم يحدد و يطبع فيما إذا كان هذا العدد زوجياً أم فردياً .

```
# include < iostream.h>
main ()
{
    int a ;
    cout <<"enter a ="; cin>>a;
    if ( a % 2 = = 0)
        cout << " not odd" ;
    else
        cout << " odd" ;
    return 0 ;
}
```

```
Inactive C:\TCWIN45\BIN \ NONAME06.EXE)
enter a = 13
odd
```

ويمكن استخدام البني if / else المتداخلة من أجل القيام بفحص عدة حالات من خلال وضع البني if / else داخل بعضها البعض . على سبيل المثال إذا كانت علامة الفحص أكبر أو يساوي 90 فيتم طباعة الحرف a وإذا كانت بين 89 و 80 فتطبع الحرف b وإلا فيتم طباعة الحرف c . وبالتالي تكون العملية C++ المكافئة بالشكل:

```
if ( grad > = 90 )
cout << "a" ;
else if ( grad >= 80)
cout << "b" ;
else
        cout << "c" ;
```

ملاحظة :

عادة تضــع تعليمة واحدة في جســم البنية الاختيارية if ولكن إذا أردنا وضــع عدة تعليمات يجب أن نقوم بوضـعها داخل قوسـين كبيرين ({ }) . نسـمى مجموعة التعليمات المحتواه ضـمن زوج من الأقواس الكبيرة بالتعليمية المركبة compound statement .

مثال 1:

```
if (grad > = 60 )
cout << " passed" ;
else
{
      cout << " failed " ;
     cout << " you must take this course again" ;
}
```

في هذه الحالة إذا كانت قيمة gradأصــغر من 60 عندها يقوم البرنامج بتفيذ التعليمتين الموجودتين في الجزء else ويطبع ما يلي:

```
failed
you must take this course again
```

<u>بعض الأمثلة:</u>

1- أكتب برنامج يأخذ كدخل عددين صحيحين من لوحة المفاتيح ويفحص فيما إذا كان الثاني قاسم للأول.

```
# include<iostream.h>
main ( )
{
int  a , b ;
cout<<"enter a=";cin>>a;
cout<<"enter b=;cin>>b;
if ( b! = 0 && a % b = = 0 )
```

```
cout << a << ' is divisible by " <<b ;
else
cout <<a<<is not divisible by " << b ;
return 0 ;
}
```

```
Inactive C\TCWIN45\BIN\NONAME00.EXE)
enter a = 25
enter b = 5
25 is divisible by 5
```

2- أكتب برنامج يأخذ كدخل ثلاث أعداد صحيحة ثم يطبع أصغر هذه الأعداد.

```
# include < iostream.h>
main ()
{
  int a , b, c ;
  cin >> a >> b >> c ;
  if ( a > b )
        if (a < c ) cout << " min is" << a ;
        else cout << " min is " << c ;
  else
        if ( b < c ) cout << "min is " << b ;
  else
        cout << " min is " << c ;
  return 0;
```

```
}
Inactive C:\TCWIN45\BIN\NONAME01.EXE)
enter  a = 10
enter b = 8
enter c = 77
min is 8
```

---

2. البنى التكرارية :

- **البنية التكرارية While** :

تسمح البنية التكرارية للمبرمج بتحديد مجموعة من الأفعال يجري تكرارها طالما ظل الشرط المرافق للبنية محققاً ، ولها الشكل العال التالي :

```
while (condition )
statement
```

مثال 1:

أكتب برنامج لطباعة الأعداد من 10−1 بشكل عمود واحد.

```
# include < iostream.h>
main ()
{
int i ;
i = 1 ;
while ( i <=10)
{
cout << i << "\n" ;
i = i +1 ;
```

```
}
return 0 ;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME02.EXE)
1
2
3
4
5
6
7
8
9
10
```

مثال 2:

يفرض لدينا علامات مذاكرة قام بها طلاب صـف مؤلف من عشـرة طلاب والمطلوب حسـاب معدل علامات طلاب الصف في هذه المذاكرة .

```
# include < iostream.h>
main ()
{
float mark , sum ;
int  i = 1
sum = 0
while ( i < = 10 )
{
```

```
cout<<"enter the mark=";cin>>mark;
sum = sum + mark ;
i = i +1 ;
}
cout<<"average is : "<<sum/10 ;
return 0 ;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME03EXE)
enter the mark = 13
enter the mark = 44
enter the mark = 54
enter the mark = 60
enter the mark = 90
enter the mark = 33
enter the mark = 75
enter the mark = 56
enter the mark = 55
enter the mark = 78
    average is : 55.8
```

ملاحظة:

1- إن عدم وضع تعليمة أو فعل جسم البنية while يسبب عدم تحقق الشرط المرافق لها وينتج عن ذلك عدم إنتهاء التكرار .

2- تسبب كتابة الكلمة while مع حرف كبير في البداية خطأ وذلك على اعتبار أن لغة C++ حساسة لحالة الحروف تحتوي كافة الكلمات المفتاحية الخاصة بلغة C++ مثل if , while ، .. وغيرها على شكل حروف صغيرة .

3- إن أي متحول لا يعطي قيمة ابتدائية يمكن أن يكون له قيمة ما لا يعرف عنها شيء مخزنة مسبقاً في موضع الذاكرة المخصص لهذا المتحول ، وبالتالي إن عدم إعطاء متحول حساب مجموع مثل sum أو عداد مثل I سوف يؤدي إلى الحصول على نتائج قد تكون خاطئة.

• البنية التكرارية do / while :

تشبه بنية التكرار do / while البنية while حيث نقوم بالبنية while بالتحقق من صحة شرط الاستمرار بالتكرار في بداية الحلقة قبل تنفيذها ، أما في حالة البنية do / while فيتم ذلك بعد تنفيذ جسم الحلقة أولاً . أي يتم تنفيذ جسم البنية do / while مرة واحدة على الأقل. عند الإنتهاء من تنفيذ البنية do / while يتم الانتقال إلى التعليمية التي تلى مباشرة جزأها while ، ولها الكل العام التالي:

```
do
  statement ;
while ( condition ) ;
```

وقد تم استخدام الأقواس الكبيرة لتحديد جسم البنية do / while حتى لا يتم الخلط بين البنيتين while , do / while ، لذلك يتم عادة كتابة البنية do / while على الشكل التالي:

```
do {
  statement
} while ( condition )
```

مثال 1:

نفس المثال السابق . أكتب برنامج لطباعة الأعداد من 1-10 بشكل عمود واحد . ولكن باستخدام do / while

```
# include <iostream.h>
main()
{
int i ;
i = 1
```

```
 do {
 cout <<i<<"\n";
 i=i+1;
     } while ( i <=10);
 return 0;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME02.EXE)
1
2
3
4
5
6
7
8
9
10
```

* بنية التكرار for:

تدعى هذه البنية أيضاً بالبني التكرارية ذات العداد ، وهي تتطلب ما يلي:

1- تعريف متحول التحكم بالحلقة (وهو عداد الحلقة)

2- تحديد القيمة الابتدائية لمتحول التحكم بالحلقة .

3- تحديد أسلوب الزيادة (أو الانقاص) الذي يتم من خلاله تغيير قيمة متحول التحكم بالحلقة في كل مرة نمر فيها.

4- تحديد الشرط الذي من خلاله نقوم بفحص النتيجة النهائية لمتحول التحكم بالحلقة (حتى نحدد إذا كان من الممكن معاودة تنفيذ الحلقة).

ولها الشكل العام التالي:

```
for ( exp 1;     exp 2  ; exp 3 )
statement ;
```

exp 1 : يمثل تعريف وتحديد القيمة الابتدائية لعداد الحلقة.

exp 2 : يمثل شرط إنهاء الحلقة أي شرط فحص النتيجة النهائية لعداد الحلقة.

exp 3 : يمثل أسلوب زيادة أو إنقاص عداد الحلقة.

مثال1:

نفس المثال السابق . أكتب برنامج لطباعة الأعداد من 1-10 بشكل عمود واحد . ولكن بإستخدام البنية for .

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

```
(Intactive C:\TCWIN45\BIN\NONAME02.EXE)
1
2
3
4
5
6
7
```

```
8
9
10
```

مثال 2:

أكتب برنامج لحساب مجموع جميع الأعداد الصحيحة من 2 إلى 100

```
# include < iostream.h>
man ( )
{
inst sum = 0 ;
for ( int i = 2 ; i < = 100 ; i = i +1)
sum  = sum + i ;
cout << " sum is " << sum ;
return 0 ;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME02.EXE)
Sum is 5049
```

عمليات الإسناد:

يتوفر في لغة C++ عدداً في من عمليات الإسناد المختصرة التي هي تعبير على عملية الإسناد نفسها، فعلى سبيل المثال يمكن اختصار التعليمية التالية:

```
c=c+3;
```

لتصبح بالشكل التالي:

```
c+=3;
```

حيث نسمي العملية =+ بعملية الإسناد والجمع addition assignment operator

وبين الجدول التالي عمليات الإسناد الحسابية مع أمثلة وشروح لها.

عملية الإسناد	مثال	الشرح
+=	c+=10	c=c+10
-=	c-=10	c=c-10
*=	c*=10	c=c*10
/=	c/=10	c=c/10
%=	c%=10	c=c10%

عمليات الزيادة بواحد والإنقاص بواحد :

يتوفر أيضـــاً في لغة c++ عملية الزيادة بواحد الأحادية unary increment operator(++) وعملية الإنقاص بواحد الأحادية unary decrement operator (--) ويلخص الجدول التالي كيفية استعمالهما :

العملية	التسمية	مثال	الشرح
++	عملية الزيادة بواحد أمامية	++a	زيادة قيمة a بواحد ثم استخدام القيمة الجديدة
++	عملية الزيادة بواحد خلفية	a++	زيادة قيمة a بواحد بعد استخدام القيمة القديمة
--	عملية إنقاص بواحد أمامية	--b	إنقاص قيمة b بواحد ثم استخدام القيمة الجديدة
--	عملية إنقاص بواحد خلفية	b--	إنقاص قيمة b بواحد بعد استخدام القيمة القديمة

مثال توضيحي :

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

```
int c ;
c = 3;
cout << c << "\n" ;
cout << c ++ <<"\n" ;
cout << c << " \n" ;

c = 3
cout << c << "\n" ;
cout << c ++ <<"\n" ;
cout << c << " \n" ;
return 0;
}
```

وتكون نتائج هذا البرنامج هي:

```
3
3
4

3
4
4
```

مثال :

أكتب برنامج يلخص نتائج امتحان مادة ما لعشــرة طلاب وذلك بعد أن أعطيت قائمة بأســماء الطلاب ومقابل كل اسم تم وضع القيمة 1 إذا كان الطالب ناجح والقيمة . إذا كان الطالب راسب في الامتحان

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

```
    int r , p, f ;
    p = 0 ; f = 0 ;
    for ( int i = 1 ; i < = 10 ; i ++ )
    {
        cout << " enter result : "; cin >> r ;
        if ( r = = 1 )
                p + = 1 ;
        else
                f+= 1 ;
    }
    cout << " passed : " <<p <<"\n" ;
    cout << " failed : ' << f << "\n" ;
    return 0 ;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME03EXE)
        enter result      : 1
        enter result      : 1
        enter result      : 1
        enter result      : 0
        enter result      : 1
        enter result      : 0
        enter result      : 0
        enter result      : 1
        enter result      : 1
        enter result      : 0
        passed : 6
        failed    : 4
```

* بنية الاختيار المتعدد switch :

يمكن أن تصادفنا حالة خاصة في إحدى البرنامج تحتوي على سلسلة من القرارات التي تتعلق بنتائج متعدد لفحص قيمة متحول أو تعبير ما ، ويمكن أن تؤدي كل نتيجة من هذه النتائج إلى القيام بفعل مختلف عن الآخر . لذلك توفر لغة C++ البنية switch من أجل التعامل مع حالات اتخاذ القرار المتعلقة بعد اختيارات ، ولها الشكل العام التالي:

```
switch ( expression )
{
  case  constant 1 : statement 1 ;
  case  constant 2 : statement 2 ;
  case  constant 3  : statement 3 ;
  case  constant 4  : statement 4 ;
  .
  .
  case constant n :  statement n ;
  default : statement 0 ;
}
```

مثال 1:

أكتب برنامج لإعطاء اسم اليوم من أيام الأسبوع عند إعطاء رقمه.

```
# include < iostream.h>
main ()
{
int c ;
cout << "enter number : " ;
cin >> c ;
switch (c )
{
```

```
case 1 : { cout << " saturday " ; beak ; }
case 2 : { cout << " sunday " ; beak ; }
case 3 : { cout << " monday " ; beak ; }
case 4 : { cout << " tuesday " ; beak ; }
case 5 : { cout << " wednesday " ; beak ; }
case 6 : { cout << " thursday " ; beak ; }
case 7 : { cout << " friday " ; beak ; }
default : { cout << " that number is out of range " ; }
}
return 0 ;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME07.EXE)
        enter number   : 7
        friday
```

مثال2:

أكتب برنامج يقوم بقراءة عددين ومن ثم يعطي ناج جمعها وطرحهما وضـربهما مسـتخدماً لعرض ذلك شـاشـة خيارات.

```
# include < iostream.h>
main ( )
{
int  n , x, y ;
cout << "1:  جمع العددين " ; cout <<"\n";
cout << "2 :  طرح العددين " : cout <<"\n";
cout <<3:  ضرب العددين" ; cout <<"\n";
cout << ******************" ; cout <<"\n";
```

```
cout <<  أدخل العدد الأول " ; cin >>x; cout <<"\n";

cout <<  أدخل العدد الثاني " ; cin >> y ; cout <<"\n";

cout <<  أدخل رقم الخيار" ; cin >> n ; cout << "\n";

while ( n!=0)

{

switch ( n )

{

case 1:

{ cout << x+y ; break ; }

case 2:

{ cout << x-y ; break ; }

case 3 :

( cout << x*y; break; }

default :

{ cout << "  الرجاء إدخال أحد أرقام الخيارات المتاحة" ; cin>>n;}

        }

}

return 0;

}
```

أمثالة عامة:

1- أكتب برنامج لقراءة ثلاث أعداد a, b, c ثم التحقق هل تصلح هذه الأضلاع لأن تكون أضلاع مثلث أم لا ، وبمعنى آخر هل يمكن أن نجد مثلث أطوال أضلاعه هي a, b, c .

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

```
# include < math.h>   //  int abs (int)  الملف الرأسي الحاويعلى جميع التواب الرياضية وتم استخدامه
                                                من أجل التابع
main ( )
{
  int  a , b, c ;
  cout << " a : " ; cin >> a ;
  cout << " b : " ; cin >> b ;
  cout << " c : " ; cin >> c ;
  if ((a+b>c) && (abs(a-b)<c)&&(b+c>a)&&(abs(b-c)<a) &&(a+c>b) && (abs (a-c)<b))
  cout << " triangle " ;
else
  cout << " not triangle " ;
return 0;
}
```

```
  (Inactive C:\TCWIN45\BIN\NONAME00.EXE)
        a : 5
        b : 4
        c : 3
        triangle
```

2. أكتب برنامج لحساب n!

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

```
main ( )
{
int n ;
double fact = 1 ;
cout << " enter value n: " ; cin >> n;
if ( n = = 0 )
cout << " n! = 1;
else
{
for ( int i = 1 ; i < = n ; i ++)
fact * i  ;
cout << " n ! = " << fact ;
}
return 0 ;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME01.EXE)
        enter value n : 5
        n * = 120
```

---

3. برنامج إيجاد قواسم عدد X

الحل:

إذا فرضـنا أن العدد 30= x فإنا نختبر الأعداد التي قبل x بحيث إذا كان باقي القسـمة عليها يسـاوي الصـفر عندئذ يكون العدد قاسما للعدد x .

```
# include < iostream . h >
main ()
{
  int  x  ;
  cout << " enter number : " ; cin >> x ;
  for ( int i = 1 ; i < =x ; i ++)
  if ( x % i = = 0 )
  cout << i << " \n";
  return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME02.EXE)
        enter number : 30
        1
        2
        3
        5
        6
        10
        15
        30
```

4. برنامج يقوم بقراءة عدد ما x ومن ثم يحدد هل هذا العدد أولي أم لا.

ملاحظة للحل :

1- لا يوجد في لغة الـC++ نمط بولياني لذلك ننشئ نمط من خلال النمط التعدادي enum.

2- لحل هذه المســـألة يلزمنا متحول اختبار f من نوع Boolean ففي البداية نســـند القيمة false إلى هذا المتحول أي نفرض أن العدد ليس أولي ، ومن ثم نبحث هل هناك عدد يقسم x وفي حال وجوده نسند لـf القيمة True . وفي النهاية نختبر قيمة المتحول f وأعتماداً عليه نحدد هل العدد أولي أم لا.

```
# include < iosteam.h >
enum boolean {true, false };                    // التصريح عن نمط تعدادي
main ( )
{
boolean f = false ;
int x ;
cout <<' enter number: " ; cin >> x ;
for ( int i = 2 ; i <x ; i ++)
  if ( x % i = = 0 )
  f = true ;
if ( f = = false )
  cout << " the x number is primary " ;
else
  cout << " the x numbe is not primary " ;
return 0 ;
}
```

```
(inactive c:\tcwin45\bin\noname03.exe)
        enter number : 67
        the  x number is primary
```

5. أكتب برنامج لحساب الحدود العشرة الأولى لهذه السلسة :

$$z = 1 - \frac{1}{1} + \frac{1}{2} - \frac{1}{3} + \frac{1}{4} - \ldots\ldots\ldots\ldots\ldots\ldots$$

```
# include < iostream.h>
# include < math.h>
main ( )
{
  int  n ;
  float z = 1;
  cout << " enter n: ' ; cin >> n;
  for ( int i = 1 ; i <n ; i ++ )
  if ( i % 2 = = 0 )
  z+= pow(i , -1) ;              // math تابع الرفع لقوة ويوجد في الملف
  else
  z-=pow (i , -1 ) ;
  cout << " z = " << z ;
  return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME04.EXE)
        enter  n : 15
      z = 0.341295
```

6. أكتب برنامج لإيجاد القاسم المشترك الأعظم لعددين وذلك باستخدام طريقة إقليدس التي تتلخص كما يلي:

أقوم بطرح العدد الأصغر من العدد الأكبر وأجعل حاصل الطرح مكان الأكبر حتى تصبح القيمتين متساويتين فتكون قيمة التساوي هذه هي القاسم المشترك الأعظم GCD .

مثال : العددين 15 و 20

	20	15
	5	15
	5	10
القاسم المشترك الأعظم	5	5

```
#include < iostream.h>
main ( )
{
int x , y ;
cout << "enter x : " ; cin >> x ;
cout << " enter y : " ; cin >> y;
while ( x! = y )
{
if ( x > y )
x - = y ;
else
y - = x ;
}
cout << " the gcd is " << x ;
return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME05.EXE)
```

enter x : 10

enter y : 35

the gcd is 5

---

7. أكتب برنامج لقراءة n عدد ثم حساب مجموع هذه الأعداد ومتوسطها وأكبر وأصغر عدد فيها: ملاحظة:

دائماً لحساب أكبر أو أصغر عدد من بين مجموعة أعداد ، نفرض أن العدد الأول هو الكبير ثم نختبر باقي الأعداد وكلما ظهر عدد أكبر جديد نجعله هو العدد الأكبر ، وهكذا حتى تنتهي مجموعة الأعداد . (بالنسبة للعدد الأكبر).

```
# include < iostream.h>
main ( )
{
int n , x , sum , max , min ;
cout << " enter n : " ; cin >> n;
cout << " enter the first number : " ; cin >> x ;
sum = x ; min = x ; max = x ;
for ( int i = 2 ; i < = n ; i ++)
{
cout << " enter number : " ; cin >> x ;
sum + = x ;
if ( x > max ) max = x ;
if ( x < min ) min = x ;
}
cout << " sum is " << sum << "\n" ;
cout << " avg is " << ( float ) sum /n << "\n" ;
cout << " max is " << man << " \n" ;
```

```
cout << " min is " << min << "\n" ;
return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME06.EXE)
        enter  n : 4
        enter the first number : 22
        enter number : 13
        enter number : 24
        enter number : 44
        sum is 103
        avg is 25.75
        max is 44
        min is 13
```

---

8. أكتب برنامج لقراءة عدد ما والتحقق فيما إذا كان عدم تام أم لا .

الحل :

نقول عن عدد ما أنه عدد تام إذا كان مجموع قواسم هذا العدد (ما عدا العدد نفسه) يساوي العدد نفسه .

مثال :

العدد 6 هو عدد تام لأن مجموع قواسم العدد 6 تساوي 06 (3+2+1=6)

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

```
main ( )
{
int x ;
int sum = 0 ;
cin>> x ;
for ( int i = 1 ; i < x ; i ++)
if ( x % i = = 0 )
sum + = i ;
if ( sum = = x )
cout << " perfect " ;
else
cout << " not perfect " ;
return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME07.EXE)
        enter number = 28
       perfect
```

---

9. أكتب برنامج لإيجاد جميع الأعداد التامة ضمن مجال [1..n]

```
# include < iostream.h>
main ( )
{
  int n , sum = 0 ;
  cin>>n ;
  for ( int i = 1 ; i <=n ; i ++ )
          {
```

```
for ( int j = 1 ; j < i ; j++)
if ( i % j = = 0 )
sum + = j ;
if ( sum = = i )
cout << "  " <<i << endl;
sum = 0 ;
}
return 0;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
        enter  n : 200
       6
       28
```

10. أكتب برنامج لإيجاد المضاعف المشترك الأصغر لعددين:

```
# include < iostream.h>
main ( )
{
      int x , y ;
      cout << " x = " ; cin >> x;
      cout << " y = " ; cin >> y ;
      if ( x > = y)
      {
            for ( int j = x ; j < x ; j++)
                  if ( j % x = = 0 ) && ( j % y = = 0 )
```

```
                    { cout <<j ; break ; }
      }
  else
     {
        for  ( int j = y ; j < x *y; j++)
              if (( j% x = = 0) && ( j % y = = 0 ))
                    { cout << " " << j ; break ; }
     }
  return 0 ;
}
```

11. أكتب برنامج لقراءة عددين والتحقق فيما إذا كانا عددين صديقين أم لا .

الحل :

نقول عن عددين أنهما صـدقيقين إذا كان مجموع قواسـم العدد الأول (ما عدا العدد نفسـه) يسـاوي العدد الثاني والعكس بالعكس.

```
# include<iostream.h>
main ( )
{
   int x , y , i ;
    int sum 1 = 0 , sum 2=0
cout <<"x="; cin>>x;
cout <<" y=" ; cin >> y;

for ( i = 1 ; i < x ; i ++ )
```

```
    if ( x % i = = 0 )
sum 1 + = i ;
for ( i = 1 ; i < y ; i ++)
     if ( y % i  = = 0 )
sum 2 + = i ;

if ( sum 1 = = y && sum 2 = = x )
cout <<x<<" friend " <<y;
else
cout << x << " not friend " << y;
reterun 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME02.EXE)
        x = 20
       y = 34
       20 not friend  34
```

وظيفة :

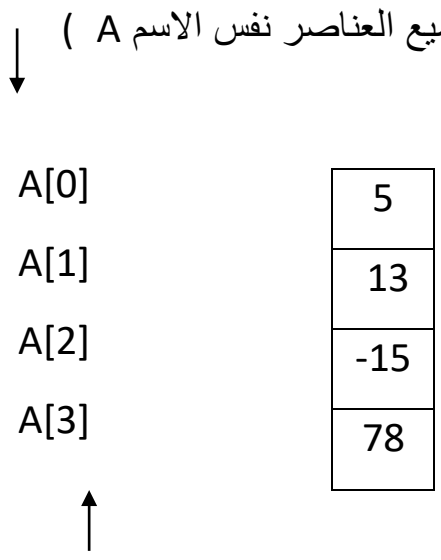
أكتب برنامج لإيجاد جميع الأعداد الصديقة ضمن مجال [1.. n] .

4-المصفوفات

المصفوفات:

المصفوفة هي عبارة عن مجموعة من خانات الذاكرة المتتالية التي لها نفس الاسم ونفس النمط . ومن أجل الرجوع الى خانة معينة من هذه الخانات ضمن المصفوفة ورقم موضع الخانة (العنصر) ضمن المصفوفة وذالك داخل قوسين متوسطين من الشكل ([ ]) .

والشكل التالي يمثل مصفوفة من الإعداد الصحيحة التي أسمها A وهي تتضمن أربعة عناصر .



{رقم موضع العنصر من عناصر المصفوفة (رقم موضع الخانة)}

العنصـر الاول من المصـفوفة هو دائماً العنصـر ذو الرقم صـفر وبالتالي يتم الرجوع اليه من المصفوفة A مثلاً على الشكل التالي A[0] وبشكل عام نستطيع القول أننا نرجع الى العنصر ذو الرقم I بكتابة [I-1] اسم المصفوفة

نسمي رقم الموضع الذي نضعه ضمن قوسين متوسطين بالدليل Subscript ويجب أن يكون الدليل عبارة عن عدد صـحيح أو تعبير يعطي قيمة صـحيحة حيث يتم حسـاب قيمة التعبير أولاً من أجل تحديد المطلوب ، على سـبيل المثال c=2 وb=3 وبالتالي يكون العنصـر A[b+c] يمثل العنصر A[5] .

- **<u>التصريح عن المصفوفات :</u>**

تشغل المصفوفات أجزاء محددة من الذاكرة لذلك نقوم بتحديد نمط عناصر المصفوفة وعددها الى المترجم الذي يقوم بدوره بحجز الحجم المناسب في الذاكرة . وبتصريح الشكل العام التالي:

[عدد عناصر المصفوفة] اسم المصفوفة نمط معطيات المصفوفة

مثلاً : int A[5]

يمكن حجز أمكنة لعدة مصـفوفات باسـتخدام تصـريح وحيد ، فعلى سـبيل المثال ; Y[13] , int x[10]

ويمكن التصريح عن المصفوفات تحتوي معطيات من أنماط أخرى مثل ; float x[100] ، ; char y [100] ، .....

ويسمى هذا النوع من المصفوفات بالمصفوفات ذات البعد الواحد .

- **<u>أمثلة عن طرق اعطاء قيم ابتدائية لعناصر المصفوفة :</u>**

1. يمكن اعطاء قيمة ثابتة لكامل العناصـر المصـفوفة فعلى سـبيل المثال نعطي الصـفر لكامل عناصر المصفوفة على الشكل التالي int A[10]={0}.

2. يمكن اعطـاء قيم ابتـدائيـة لعنـاصـــر المصـــفوفـة أثنـاء التصـــريح عنهـا مثلا ;
int A[5]={10,2,34,6,18}

3. يمكن أعطاء قيم ابتدائية لعناصر المصفوفة بالشكل التالي :

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

مثال 1 :

أكتب برنامج يقوم بطباعة عناصر مصفوفة .

```
# include < iostream.h >
main()
{
int a[5] = {10,2,12,30,67} ;
for (int i =0; i<5 ;i++)
     cout <<"a["<<i<<"] = "<<a[i]<<"\ n ";
```

```
return;
}
```

ملاحظات :

1 ـ يسبب التصريح التالي :

```
int [5] ={1,2,34,56,24,14};
```

خطأً قواعدياً لاننا أعطينا ستة قيم لمصفوفة مؤلفة من خمس عناصر فقط .

2 ـ يسبب التصريح التالي :

```
int n[5]={1,2,9,5};
```

اعطاء قيمة الصفر للعنصر الخامس من قبل المترجم .

3 ـــ اذا تم حذف حجم المصفوفة أثناء التصريح عنها فان عدد عناصر هذه المصفوفة يصبح مساوياً لعدد القيم الابتدائية المعطاة ضمن القائمة الملحقة بالتصريح . لذلك يقوم التصريح التالي :

```
int n[ ] = {1,2,3,4,5,6} ;
```

بخلق مصفوفة مؤلفة من ستة عناصر .

التصريح عن متحول ثابت :

يكون الشكل العام للتصريح عن المتحول ثابت كالتالي :

Const نوع المعطيات اسم المتحول = القيمة ;

مثال :

Const int size=10;

يفيد السطر السابق في التصريح عن متحول ثابت Size وذلك باستخدام الكلمة المحجوزة const 10

ملاحظة هامة :

يجب اعطاء قيمة ابتدائية للمتحولات الثابتة عند التصريح عنها **ولا يمكن تغيير هذه القيمة بعد ذلك** ، تسمى المتحولات الثابتة أيضاً بالثوابت constants .

ومن **الاخطاء البرمجية الشائعة** اعطاء قيمة لثابت من خلال تعليمة تنفيذية مثل :

```
main ( )
{
   const int n ;
  n = 9 ;
   return 0 ;
}
```

وبالتالي تعطي التعليمات السابقة خطأً قواعدياً نتيجة اسناد متحول ثابت ، ويكون التصحيح كما يلي :

```
# includde < iostream. h>
main ()
{
  const int n = 9
```

```
    cout <<" the value of constant is : "<< n ;
    return 0 ;
}
```

- **المصفوفات والثوابت :**

يمكن وضـــع المتحولات الثابتة في أي مكان يمكن أن نضـــع فية تعبيراً ثابتاً ، فمثلاً يمكن استخدامها في تحديد حجم المصفوفة .

مثال :

```
contest int size = 10 ;

int s [size ] ;
```

تفيد التعليمات السابقة في تحديد حجم مصفوفة S باستخدام الثابت SIZE .

ويفيد اسـتخدام المتحولات الثابتة لتحديد حجم المصـفوفات في جعل البرامج أكثر قابلية لتغيير الحجم . فمثلاً حلقة FOR تقوم بتعبئة 10 عناصـــر يمكن تعديلها لتقوم بتعبئة 1000 عنصـــر وذلك بتغيير قيمة الثابت المرتبطة به أما في حالة عدم أستخدام الثوابت فيتطلب التعديل السابق عدة تعديلات في أماكن مختلفة من البرنامج .

مثال 1:

اكتب برنامج لطباعة عناصر مصفوفة .

```
# include< iostream.h>
main( )
{
  const int arrasize = 10 ;
  int a [arrasize];
```

```
  for (int i=0 ;i<arrasize ; i++)
  {
    a[i]=2+2*i;
     cout<<a[i]<<"\n";
   }
    return 0;
}
```

مثال 2 :

أكتب برنامج لحساب مجموع عناصر مصفوفة .

```
# include< iostream.h>
main( )
{
   const int arrasize = 10 ;
    int a [arrasize]={1,12,5,4,8,9,7,32,65,91};
    int sum =0;
    for (int I =0;i< arrasize ; i++)
        sum +=a[i];
    cout << "sum = "<<sum;
    return 0;
 }
```

- **مصفوفات الحروف:**

سوف نتعرض الان الى تخزين سلاسل الحروف فى مصفوفات من النمط Char حيث أن أي سلسلة حروف مثلا السلسلة "first" هى فى الواقع عبارة عن مصفوفة حروف . يمكن إعطاء قيمة ابتدائية لمصفوفة حروف باستخدام سلاسل الحروف فعلى سبيل المثال يقوم التصريح بالشكل التالى

Char str 1[ ] =" first "

بإعطاء قيم ابتدائية لكل عنصر من عناصر المصفوفه str1حيث يقابل كل منها احد حروف السلسلة "first" ويتحدد عدد عناصر المصفوفة str1 بواسطة المترجم وذلك حسب طول السلسلة المعطاة . من المهم أن نلاحظ أن السلسلة "first" تحتوى على خمسة حروف إضافة الى حرف خاص يحدد نهاية السلسلة وهو الحرف الصفري null character لذلك تتألف المصفوفة str1 من ستة عناصر ويتمم تمثيل الحرف الصفري على الشمل '0 \'. وهذا يعنى أن كافة الحروف تنتهى بالحرف الصغرى ويتم بالتالى التصريح عن المصفوفات التى تتعامل مع هذه السلاسل بحيث تكون ذات حجم كافي لتخظين حروفها إضافة الى الحرف الصفري

يمكن أيضا إعطاء قيم ابتدائية لمصفوفات الحروف باستخدام ثوابت الحروف المفردة ضمن قائمة للقيم الابتدائية . فمثلا يمكن كتابة التصريح السابق على الشكل التالى

```
char str1 [ ] ={‘ f ‘,’ i ‘,’ r ‘,’ s ‘,’ t ‘,’\0’};
```

وعلى اعتبار ان سلاسل الحروف هي عبارة عن مصفوفات للحروف فيمكن الوصول الى كل حرف من حروفها بشكل منفصل مباشرة باستخدام ذليل عناصر المصفوفة فعلى سبيل المثال يمثل العنصر str1[0] الحرف f' ويمثل الحرف t' العنصر str1 [4]

يمكن ايضا إدخال السلاسل مباشرة الى مصفوفات الحروف باستخدام لوحة المفاتيح وذلك بواسطة cin >> فمثلا التصريح التالى

Char str2 [10] ;

يقوم بإنشاء مصفوفة حروف قادرة عل تخزين سلسلة من 9 أحرف والحرف الصفري ايضا . وتمكن التعليمة التالية :

cin >>Str2;

على قراءة سلسلة من الحروف من لوحة المفاتيح وتخزينها فى str 2 أما التعليمة التالية

```
cout >> Str2 ;
```

فتساعد على طباعة المصفوفة str2

ملاحظة:

عند قراءة سلسلة حروف من لوحة المفاتيح لم يتم كتابة حجم المصفوفة وإنما فقط إسمها وبالتالى فى حالة عدم التزويد بمصفوفة ذات حجم كافي لاستيعاب الحروف المدخلة من قبل المستخدم بواسطة لوحة المفاتيح تؤدى الى ضياع فى معطيات البرنامج بالاضافة الى اخطاء التنفيذ علما أن cin يقوم بقراءة الحروف المدخلة حتى يصل الى فراغ ولا يهتم بحجم المصفوفة وكذلك الطباعة coutلاتهتم بحجم المصفوفة ويتم طباعة الحروف حتى الوصول الى الحرف الصفري .

مثال توضيحى :

```
# include <iostream.h >
main ( )
{
char str1[10],str2[]="first program";
cin>>str1;
cout<<"str1:"<<str1<<"\n"<<"str2:"<<str2<<"\n";
for(int i=0;str[i]!="\0";i++)
cout<<str[i]<<" ";
Return 0;
}
```

hello there

str 1 : hello

str 2 : first program

h e l l o

فرز المصفوفات :

تعتبر عملية فرز المعطيات (أي وضعها حسب ترتيب معين تصاعدي أو تنازلى مثلا) من أهم التطبيقات الحسابية وبالتالى سوف نقوم بشرح طريقة فرز تدعى بالفرز الفقاعي bubble sort او الفرز بالغوص sinking cort وذلك لان القيم الصغيرة تقوم تدريجيا بشق طريقها تصاعديا الى قمة المصفوفة بينما تقوم القيم الكبيرة بالغوص الى اسفل المصفوفة وتعتمد هذه الطريقة في الفرز على القيام بأكثر من مرور على العناصر وفى كل مرة يتم مقارنة زوجين متتالين من عناصر المصفوفة إذا كان هذان الزوجان مرتبين تصاعديا (أو لهما نفس القيمة) فأننا ندعهما على حالهما وإذا كان مرتين تنازليا فإننا نقوم بالمبادلة بينعما ضمن المصفوفة

يقوم البرنامج التالي بمقارنة العنصرين a[0] و a [1] ثم العنصرين a[1] و a [2] وهكذا حتى نهاية المصفوفه بمقارنة العنصرين a[8] و a [9] وعلى اعتبار أن المصفوفة تحتوى على عشرة عناصر فالبرنامج يقوم بتسع مقارنات تشق خلالها القيمة الكبرى طريقها الى الاسفل بينما تصعد القيمة الصغرى مكانا واحدا وهذا يعنى أن القيمة الكبرى سوف تصل الى الموضع a [9] بعد نهاية المرور الاول أما القيمة الكبرى الثانية سوف تصل الى الموضع

a[8] بعد نهاية المرور الثاني وهمذا حتى المرور التاسع حيث توضع القيمة التاسعة فى الموضع a[1] ويؤدى ذلك لبقاء القيمة الصغرى فى الموضع a[0] إذا نحتاج الى تسعة مرورات لفرز مصفوفة مولفة من عشر عناصر

تتم عملية الفرز من خلال بنية التكرار for المتداخلة وتجرى عملية المبادلة بين العناصر وفقا للتعليمات التالية

hold = a [i] ؛

```
a [i]  = a [i+ l] ;

a [i+ l] = hold ;
```

ونستخدم المتحول الاضافي Hold لتخزين إحدى القيمتين المراد مبادلتهما مؤقتا

```
a [ i ] = a [i+1];

a[ i +1 ] = a[ i];
```

فإذا كانت القيمة a[i] تساوى 10 وقيمة a[i+l] تساوى 8 فإن التعليمة الاولى تجعل قيمة العنصرين مساوية للقيمة العنصرين مساوية للقيمة 8 مما يسبب ضياعا للقيمة 10

```
#include < iomanip.h >
main ( )
{
const int size = 10 ;
int a[ size] ; int hold ;
for ( int i = 0; i< size ; i++)
{
cout << setw (5) << '' a['' << i <<''] ='' ;
cin >> a[ i] ;
cont << endi;
}
for ( int pass = 1 ; pass < size – pass ; i++ )
if ( a[ i ]> a [ i+ 1 ])
```

```
{
hold = a [ i ] ;
a [ i ] =a [ i+1 ] ;
 a[ i+1 ] = hold ;
}
for ( i=0 ; i < size ;i + + )
 cout < < setw (4) < < a[i] ;
return 0;
}
```

ملاحظة :

يتميز الفرز الفقاعي بسهولة البرمجة ولكنة أسلوب فرز بطيء وخصوصا مع المصفوفات الكبيرة

- المصفوفات المتعددة الأبعاد :

يمكن للمصفوفات فى لغة ++cأن تأخذ عدة أبعاد (بعدين وأكثر وصولا إلى 12 دليلا 9 ومن بين الاستخدامات الشائعة المصفوفات الثنائية أو الجداول التي تتألف من الأسطر والاعدمدة . وبالتالي للحصول على عنصر ما من بين العناصر يجب أن نحدد الدليلين : رقم السطر ورقم العمود الذي ينتمي لها العنصر . فمثلا إذا كان لدينا مصفوفة a مؤلفة من ثلاثة أسطر و أربعة أعمدة أي مصفوفة 3X4 فإننا نحدد كل عنصر من عناصر المصفوفة

بـ a [i] [j] حيث أن a اسم المصفوفة و I ، j هما الدليلان المحددان للعنصر المطلوب . حيث تأخذ عناصر السطر الأول القيمة صفر للدليل I أما عناصر العمود الأول فتأخذ القيمة صفر للدليل j والتالي يمثل a [0] [0] العنصر الأول من السطر الأول والعمود الأول .

صفر إعطاء قيم ابتدائية لعناصر المصفوفة المتعدد الأبعاد بنفس أسلوب المصفوفات ذات البعد فمثلا يمكن إعطاء قيم ابتدائية للمصفوفة [2] [2] a بالشكل التالي

```
int  a [2] [2] ={ {2.4} ,{5.9} };
```

حيث يتم تجميع عناصر كل سطر ضمن قوسين كبيرين . مما يدب على أن القيم 2 و 4 هي قيم العنصرين [0] [0] a و [1] [0] a والقيم 5 و 9 هي [1] [0] a a [1] [1]

```
# include < iostream.h >
# include< iomanip.h >
main( )
{
const int  size 1 = 3 ;
const int  size 2=2;
int a [ size 1 ] [ size 2] ;
for ( int i=0 ; i < size 1 ; ++ )
for ( intj=0 ; j < size 2 ;j ++ )
{
cout < < setw ( 5 ) < < "a["< <" ]["< <j< <"]=";
cin > > a[ i ] { j ] ;
cout < < endl ;
}
retum0 ;
```

في حالة كانت القيم الابتدائية غير كافية لعناصر السطر فانه يتم إعطاء القيمة صفر لبقية العناصر .

مثال : int a [2] [2] ={ {3} ,{4.6} };

يعطى العنصر [0] [0] a القيمة 3 [1] [0] a القيمة 6 أما العنصر[1] [0] a فتسند له قيمة الصفر من قبل المترجم

مثال

اكتب برنامج لقراءة عناصر مصفوفة ثنائية مدخلة من قبل المستخدم

تمارين عامة :

1- اكتب برنامج لقراءة صف ذو بعد واحد ثم طباعته على الشا شة

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int size =3;
int a[size ];
for ( int I = 0 : < size : i++ )
```

```
for (i=0;<size ;i++)
cout <<''a[''<<''a[''<<i<<'']=''<<a[i] <<setw(5);
retum 0;
}
```

2- اكتب برنامج لقراءة قيم صف ذو بعد واحد ثم احسب مجموع ومتوسط عناصر هذا الصف بالإضافة إلى اكبر واصغر عنصر

```
#inlude <iostream.h>
main ( )
const int size =3;
for ( int I = 0 : < size : i++ )
    cout <<''a[''<<i<<'']= ;
cin>>a[i];
}
int sum= 0,max =a[0],min=a[0];
for(i=0;i<size;i++)
{
sum+=a[i];
if(max<a[i])max=a[i];
if(min>a[i])min=a[i];
```

```
}
cout<<''sum is:''<<sum<< end1;
cout<<average is: ''<< sum/size << endl;
cout <<''max is:''<<max<<end1;
cout<<''min is :''<<min <<end1;
```

3- اكتب برنامج لقراءة عناصر صفين بعد أحادي ثم احسب مجموع هذين الصفين وحداؤهما

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
const int size =3;
{
int a[size],b[size],c[size];
{
cout<<''a[''<<i<<]='';
cin>>a[i];
}
for(i=0;i<size;i++)
{
```

```
cout <<''b[''<<i<<'']='';
cin>>b[i];
}
for(i=0;i<size;i++)
{
c[i]=a[i]+b[i];
cout << setw(10) <<''c[''<<'']=''<<c[i];
mul+=a[i]*b[i];
cout<<setw(15)<<''mul is:''<<end1;
retum 0;
```

4- اكتب برنامج لقراءة قيم مصفوفة ذات بعدين ثم أطبع هذه القيم حسب الشكل الرياضي المتعارف علية

مثال :

9	5	1
3	7	4
2	6	8

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
const int size 1=3;
const int size 2=4;
```

```
int a[size1][size2];
for(int i=0 ;i<size1;i++)
   for(int j =0;j<size2;j++)
{
  cout <<''a[''<<i<<''][''<<j<<'']='';
cin>>a[i][j];
}

for(int j=0; j<size2 ;j++)
     cout << setw(5) <<a[i][j];
cout<<end1;
}
return 0;
}
```

5- أكتب برنامج لقراءة قيم مصفوفة مربعة ثم احسب مجموع عناصر القطر الرئيسي ومجموع عناصر القطر الثانوي

ملاحظة :

- عناصر القطر الرئيسي هي a [i] [j] حيث i = j

- عناصر القطر الثانوي a [i] [j] حيث i+ j = n- 1 ، n بعد المصفوفة .

الحل:

```
#inlude <iostream.h>
const int size =4;
```

```
main ( )
{
        int b [size][size];
        int I,j,sum1=0,sum2=0;
        for (i=0;i<size;i++)
        for(j=0;j<size;j++)
{
cout<<'' sum master : ''<< sum1<<end1;
cout<<''sum primary :''<<sum2<<end1;
return 0;
}
```

6- اكتب برنامج لقراءة قيم مصفوفة ذات بعدين ومن ثم قراءة قيمة عددية ما والتحقق من وجودها ضمن قيم المصفوفة أم لا .

```
#inlude <iostream.h>
const int size =3;
enum bool {true, false};
main( )
                int b [size][size]; int I,j,x,
                bool f =false ;
               for(i=0;i<size;i++)
```

```
for(j=0;<size;j++)
 {
 cout <<''b[''<<i<<''][''<<j<<'']='';
  cin >>b[i][i];
```

7- أكتب برنامج لحساب منقول مصفوفة ذات بعدين

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int size =3;
          int b [size][size],c[size][size];int i , j ;
          for(i=o ;i<size;i++)
          for(j=0;<size;j++)
          {
cout<< '' b [''<<i<<''][''<<j<<'']='';
cin>>b[i][j];
}
```

```
for(i=o;i<size;i++)
   for(j=o;j<size;j++)
      c[i][j]=b[j][i];
for(i=o;<size;i++)
```

برنامج لحساب منقول مصفوفة ذات بعدين

```
for(j[O;j<size;j++)
   cout <<c[i][j];<<setw (5);
cout <<end1;
}
return 0;
```

8- اكتب برنامج لعكس قيم صف a ذو بعد واحد جديد b أي يصبح أول عنصر من a آخر عنصر من b وهكذا .

مثال a [1 , 5, 9 ,4 , 7] ⟶ b [7, 4 , 9 , 5 , 1]

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int n =5;
int a[n], b [n];
```

```
for(int i=o;i<n;i++)
{
  cout <<''a[''<<i<<'']='';
cin>>a[i];
}
for(i=o;<n;i++)
{
   b[i]=a[ (n-1) –i];
cout <<''b[''<<i<<'']=''<<b[i] <<setw(5);
}
return 0;
```

9- أكتب برنامج للتحقق من تناظر مصفوفة مربعة .

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int n =3;
int a[n] [n];
 bool f=true;
for(inti=0;i<n;i++)
```

```
   for(int j=0; j;<n;j++)
{
   cout <<''a[''<<i<<'']['']<<j<<'']='';
         }
    for( i0;<n;i++)
        for(int j=0; j;<n;j++)
            if(a[i][j]!=a[j][i])
                  f=false
    if(f= = true)
        cout <<''mathed'';
else
}
for (i=0;i<n;i++)
  for(int j=0; j<n ;j++)
       if(a[i][j]!a[i][j]
           f=false
if( f = = true )
    cout << ''mathed'';
else
   cout<<''no mathed'';
return 0;
}
```

10- نقول عن جملة أو عدد انه palindrome إذا أمكن قراءتها من البداية الى النهاية وبالعكس

مثال 12321 - 555 - 45554 - 121 - radar .....

أكتب برنامج يقوم بإدخال سلسلة (من الحروف أو من الأعداد الصحية) مؤلفة من خمس خانات كحد أقصى ويتحقق فيما إذا كان هذا العدد هو palindrome أو لا.

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
   const int n=5;
   char s [n];
   int i= o ;bool f = true;
   cin >>s;
   while(s[i]!=' 0')
{
   if(s[i]!=s[(n-1)-i])
       f=false;
    i=i+1;
}
if(f = = false )
    cout <<'' not palindrome';
```

```
else
    cout <<''palindrome '';
return0;
}
```

البرمجة الشيئية بلغة

المحتويات

اسم الفقرة **الصفحة**

السجلات

records

2-1 مقدمة:

سنعرض في هذا الفصل أسلوب التفكير باستخدام الأغراض في لغة c++ لذلك سنقوم بمراجعة سريعة للمفاهيم والمصطلحات الأساسية المستخدمة المتعلقة بهذا الأسلوب.

تستخدم البرمجة غرضيه التوجه (oop) الصفوف classes التي تتضمن المعطيات والتوابع. ويمكن للمبرمج أن ينشئ أغراضا انطلاقا من الصفوف.كما يمكن استخدام الصف عدة مرات لإنشاء المزيد من الأغراض المشتقة من نفس الصف.

تميل أساليب البرمجة بلغة cأو غيرها من لغات البرمجة الإجرائية إلى الاهتمام بالأفعال ، في حين تسمى الأساليب المتعلقة بلغةc++إلى الاهتمام بالأغراض.

تعتبر التوابع الوحدة الأساسية لبناء البرنامج بلغة c.أما في لغة c++ فيتم استخدام الصفوف التي تنسخ فيها الأغراض لبناء البرامج وتطويرها. لذلك فإن المبرمجين بلغة c++ ينصب اهتمامهم بالدرجة الأولى على إنشاء أنماط معرفة من قبلهم (الصفوف). ويتضمن كل صف المعطيات والتوابع التي يتعامل معها. تسمى المعطيات الخاصة بالصف بالمعطيات الأعضاء وتسمى التوابع الخاصة بالصف بالتوابع الأعضاء.

بنفس الطريقة التي نسمي فيها النسخة المشتقة من نمط معرف مسبقا (مثل int)متحولا فإننا نسمي أي نسخة مشتقة من صف غرضا (object) وأخيرا تعتبر الصفوف في لغة c++ تطورا طبيعيا لمفهوم السجل structفي لغة c لذلك سوف نتعرف فيما يلي على السجلات قبل الخواص بالتفاصيل المتعلقة بالصفوف.

2-2 تعريف السجلات

تعتبر السجلات نمطاً من أنماط المعطيات التي يتم بناؤها باستخدام أنماطاً أخرى

مثلا:

```
struct  time
{
        int  h ;    // 0-23
        int  m ;    // 0-59
        int  s ;    // 0-59
};        // end struct time
```

يبدأ تعريف السجل بالكلمة المحجوزة structيليها اسم السجل. أما الأسماء الموضوعة ضمن القوسين { } فتسمى بالحقول المرتبطة بالسجل ويجب أن تكون هذه الأسماء مختلفة ويمكن أن تكون أنواعها مختلفة أيضاً أو متطابقة. ويجب أن ينتهي السجل بالفاصلة المنقوطة (;) ولا يمكن للسجل أن يتضمن حقلاً من نفس نمط السجل.

لا يقوم تعريف السجل بحجز أي حيز من الذاكرة فهو ليس إلا مجرد تعريف نمط جديد من المعطيات يمكن أن يستخدم للتصريح عن متحولات بنفس الطريقة المستخدمة للتصريح عن متحولات من أنماطاً أخرى معروفة فمثلاً التصريحان التاليان :

```
time  tj;

time  tarray[10];
```

في الأول tj تم التصريح عن متحول من النمط time.وفي الثاني tarray تم التصريح عن مصفوفة مؤلفة من 10 عناصر من النمط time.

والصيغة العامة لتعريف السجل هي :

```
struct      < name struct> {
```

"أسماء المتحولات مع أنماطها"

```
};
```

تعتبر أسماء السجلات اختيارية وفي حال عدم وجودها لا يمكن التصريح عن متحولات أخرى لها نفس نمط تعريف السجل إلا ضمن التعريف نفسه ، مثلا:

```
struct {
   int h,m,s;
    }x,y,z;
```

3-2 الوصول إلى حقول السجلات :

يتم الوصول إلى حقول سجل باستخدام العملية (.) النقطة فعلى سبيل المثال طباعة قيمة الحقل h ضمن السجل tj يكون بالشكل التالي :

```
cout<<tj.h ;
```

كما نستخدم العملية (<-) (إشارة الناقص مع إشارة اكبر من) للوصول إلى سجل باستخدام مؤشر على ذلك السجل فإذا كتبنا :

```
time *tp=&tj ;
```

فانه قد تم إعطاء عنوان السجل tj كقيمة للمؤشر tpو يمكن مثلا طباعة الحقل h التابع للسجل tpبالشكل الأتي :

```
cout<<tp -> h ;
```

و العملية (<-) تكافئ (.) وهي ذات أولوية أعلى من العملية (*) التي تعطي القيمة التي يؤشر عليها المؤشر.

2-4 بناء النمط time باستخدام السجلات (struct)

البرنامج الآتي يوضح كيفية بناء نمط time (سجل) المؤلف من حقول ثلاث والذي استخدمناه في التصريح عن متحول وأسندنا له قيم وطبعناها.

```
#include<iostream.h>
struct  time {
            int h,m,s ;
            };
void  main( )
   {
     time  t;
      t.h=10;
      t.m=37;
      t.s=49;
    cout<<t.h <<":"<<t.m<<":"<<t.s<<"pm"; // `10:37:49 pm
    }
```

المثال الأتي نصرح فيه عن متحولين من النوع time احدهما ساكنt1 والثاني مؤشرt2نسند t2=&t1 ونطبع

```
#include<iostream.h>
struct  time {
int h,m,s ;
                };
void  main( )
    {
       time  t1,*t2;
        t1.h=10;
        t1.m=37;
        t1.s=49;
      cout<<t.h <<":"<<t.m<<":"<<t.s<<"pm";
        t2=&t1;
cout<<t2->h <<":"<<t2->m<<":"<<t2->s<<"pm";
}
```

2-5 تمرير السجلات إلى التوابع*

توجد طريقتان لتمرير السجلات إلى التوابع. حيث تجري عملية تمرير السجلات إلى التوابع بالقيمة بشكل افتراضي. لكن يمكن تمرير السجلات بالعناوين من طريق العملية(&).ويتم تمرير مصفوفات السجلات (مثلها مثل بقية المصفوفات) بالعنوان بشكل أوتوماتيكي.

ولتمرير المصفوفات بالقيمة يجب إنشاء سحل على أن تكون المصفوفة احد حقوله على اعتبار أن السجلات تمرر بالقيمة بالتالي سوف تمرر المصفوفة معه بالقيمة.

ملاحظات(2-1):من الخطأ الافتراض أن السجلات تشبه المصفوفات وأنها تمرر بالعنوان بشكل أوتوماتيكي.

يعتبر تمرير السجلات بالعنوان(وخصوصاً من اجل السجلات الكبيرة) فعالاً أكثر بكثير من تمريرها بالقيمة (لان ذلك يتطلب نسخ كامل السجل بعد تمريره).فمثلا برنامج السجل time السابق باستخدام دالة إدخال يصبح بالشكل:

```
#include<iostream.h>
struct  time {
int h,m,s ;
                };
void Inputtime(time &);
void  main( )
    {
      time  t;
     Inputtime(t);
     cout<<t.h <<":"<<t.m<<":"<<t.s<<"pm\n"; // 10:37:49 pm
     }
void Inputtime(time &t)
{
       t.h=10;
       t.m=37;
```

```
        t.s=49;
}
```

مثال (1-1)

. البرنامج الآتي يعمل على إدخال تاريخين مختلفين ويستدعي دالة لتحديد التاريخ الأحدث وطباعته

```
# include <iostream.h>
struct date
 {
int day;
int  month;
int  year;
  };
date    max(date , date );
void   main( )
     {
        date  d1,d2,d3;
        cout<<"inpu day      month         year \n";
        cin>>d1.day>>d1.month>>d1.year;
        cout<<"inpu day      month         year \n";
        cin>>d2.day>>d2.month>>d2.year;
         d3=max(d1,d2);
```

```
        cout<<"\n output max date :\n";
        cout<<d3.day<<"/"<<d3.month<<"/"<<d3.year;
      }
date  max(date d1,date d2)
      {
if(d1.year>d2.year) return d1;
else if(d1.year<d2.year) return d2;
else if(d1.month>d2.month) return d1;
else if(d1.month<d2.month) return d2;
else if(d1.day>d2.day) return d1;
else return d2;
        }
```

في الفقرات القادمة أمثلة على تمرير السجلات إلى التوابع .

6-2السجلات المركبة

يمكن أن يكون السجل حقلاً من سجل أخر فمثلاً السجل time يمكن أن يكون حقلاً من السجلdate.

مثال (2-1):في هذا المثال نجد أن السجل time هو حقل الرابع من حقول السجل date

```
struct  time{
```

```
        int h,m,s;
      };
struct date{
        int day;       //1-30
        int month;   //1-12
        int year;     // 1300-1500
        time  t;
      };
```

نلاحظ من المثال (2-1)أن السجل time يمثل احد حقول السجل date حيث عرفنا السجل time قبل السجل date ولبيان كيفية الوصول إلى حقول هذا السجل نأخذ المثال البرمجي الآتي :

مثال (3-1):في البرنامج الأتي نسند قيم لحقول السجل المركب date وبعدها نظهر هذه القيم التي تمثل التاريخ والوقت.

```
#include<istream.h>
struct  time{
        int h,m,s;
      };
struct  date{
        int day;
        int month;
        int
year;
```

*إن كلمتي دالة وتابع تحملان نفس المعنى

```
            time t;
           };
void  writedate (date &);
void  main( )
    {
     date d;
     d.day=23;
     d.month=9;
     d.year=1393;
     d.t.h=11;
     d.t.m=14;
     d.t.s=59;
    writedate(d);
    cout<<endl;
    }
void writedate (date &d)
     {
       cout <<"/"<<d.day <<"/"<<d.month<<"/"<<d.year<<"
"<<
            d.t.h<<":"<<d.t.m<<":"<<d.t.s<<"am";
      }
```

إن خرج البرنامج السابق هو:

/23/9/1393 11:14:59 AM

مثال(1-4)

يعمل نفس عمل التمرين السابق لكن باستخدام المؤشرات على السجلات

```
# include <iostream.h>
struct date {
int day;
int  month;
int  year;
            };
date   *max(date* , date* );
void   main( )
    {
       date  *d1,*d2,*d3;
       d1=new date;
       d2=new date;
       cout<<"inpu-d1: day      month        year \n";
       cin>>d1->day>>d1->month>>d1->year;
       cout<<"inpu-d2: day      month        year \n";
       cin>>d2->day>>d2->month>>d2->year;
```

```
        d3=max(d1,d2);
        cout<<"\n  output  max  date :\n";
        cout<<d3->day<<"/"<<d3->month<<"/"<<d3->year;
      }
date  *max(stud date *d1,stud date *d2)
      {
if(d1->year>d2->year) return d1;
else if(d1->year<d2->year) return d2;
else if(d1->month>d2->month) return d1;
else if(d1->month<d2->month) return d2;
else if(d1->day>d2->day) return d1;
else return d2;
       }
```

2 –7مصفوفة السجلات

قد نحتاج في بعض الأحيان إلى التصريح عن مصفوفة من السجلات (سجلات طلاب – سجلات موظفين مثلا) تتيح لنا لغة c التصريح عن مصفوفة سجلات، المثال الآتي توضح كيفية التعامل مع مصفوفة السجلات.

مثال (5–1):

البرنامج الآتي يعرف سجل طالب مؤلف من خمسة حقول : الحقل الأول للاسم وهو من النوع سلسلة محارف، وثلاثة حقول من النوع الأعداد الصحيحة تمثل درجات لمواد ثلاث أما الحقل الأخير فهو من النوع الحقيقي ويمثل معدل الطالب في المواد الثلاث :

```
#include<iostream.h>
#include<string.h>
#include<stdlib.h>
struct stud {
                char   name[20];
                int   a,b,c;
                float  m;
             };

void  main( )
 {
    const int  d=3;
    stud  s[d],w;
   cout<<"input  records  stud";
    for ( int i=0 ; i<d ; i++)
      {
       cout<<" input name:";
       cin>>s[i].name;
       cout<<"input  a  b  c  :";
```

```
            cin>>s[i].a>>s[i].b>>s[i].c;
            s[i].m=(s[i].a+s[i].b+s[i].c)/3.0;
          }
      for (int k=0 ;k<d ; k++)
        for(int j=0 ; j<d-k-1; j++)
              if (s[j].m>s[j+1].m)
             {
              w=s[j];
              s[j]=s[j+1];
              s[j+1]=w;
              }
      cout<<" numer  name    a    b      c      m \n";
      for(int e=0;e<d;e++)
            cout<<"   "<<e<<'-'<<s[e].name<<"      "<<s[e].a  <<"
    "<<
              s[e].b<< "      "  <<s[e].c<<"       "<<s[e].m<<endl;
}//end main
```

مثال(6-1):

نفس البرنامج السابق لكن باستخدام التوابع

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

```
#include<string.h>
#include<stdlib.h>
struct  stud {
                char   name[20];
                int   a,b,c;
                float  m;
          };
    void  readstud( stud [] ,const int);
    void  sortstud( stud [] ,const int);
    void  writestud( stud [],const int);
    void main( )
     {
                const int  d=3;
               stud  s[d];
                cout<<"input records stud";
                readstud( s,d);
                sortstud( s,d);
                writestud(s,d);
      }
    void readstud( stud s[],const int d)
          {
```

```
        for(int i=0;i<d;i++)
          {
                cout<<" input name:";
                cin>>s[i].name;
                cout<<"input  a  b  c  :";
                cin>>s[i].a>>s[i].b>>s[i].c;
                s[i].m=(s[i].a+s[i].b+s[i].c)/3;
        }
}
 void  sortstud( stud s[],const int d)
       {
        stud w;
        for(int i=0;i<d;i++)
          for(int j=0;j<d-i-1;j++)
               if(s[j].name>s[j+1].name)
                {
                      w=s[j];
                      s[j]=s[j+1];
                      s[j+1]=w;
                 }
        }
```

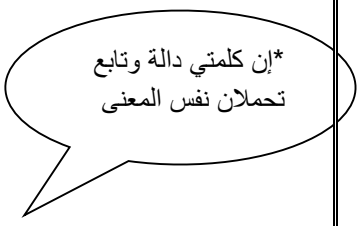
```
void writestud(stud s[],const int d)
    {
        cout<<"\n nombr  name    a    b     c         m \n";
        for(int i=0;i<d;i++)
            cout<<i<<'-'<<s[i].name<<"  "<<s[i].a<< "     "
          <<s[i].b<<"    "<<s[i].c<<"        "<<s[i].m <<"\n";

    }
```

مثال(7-1):

البرنامج الآتي يطبع الوقت بصيغتين

```
#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
struct time
{
int h;
int m;
int s;
```

```
};
void printu(const time &);
void prints(const time &);
void main()
{
    time d;
    d.h=18;
    d.m=30;
    d.s=0;
    cout<<"type 1:";
    printu (d);
    cout<<"\n type 2:";
    prints (d);
    cout<<"\n type time 3:";
    d.h=29;
    d.m=73;
    printu(d);
    cout<<endl;
    system("pause");
    return 0;
}
```



```
void printu (const time &t)
{
cout <<t.h<<":"<<t.m<<":"<<t.s;
}
void prints(const time &t)
{
cout<<((t.h==0||t.h==12)?12:t.h%12)<<":"<<t.m<<":"<<t.s
<<(t.h<12?"am":"pm");
}
```

2-8 إعطاء قيم بدائية للسجلات

يمكن إعطاء قيم بدائية للسجلات عند التصريح عنها باستخدام قائمة لهذا القيم كما هو الحال مع المصفوفات. ويتم ذلك على الشكل التالي:

```
time t={20,45,55};
```

يقوم التصريح السابق بالتصريح عن متحول سجل من النمط time ويعطى الحقل h القيمة 20 والحقل m القيمة 45 والحقل s القيمة 55 .

إذا كان عدد القيم البدائية الموجودة ضمن القائمة اقل من عدد حقول السجل فانه يتم استاد هذه القيم ألياً للحقول الأولى وتأخذ بقية الحقول القيمة الافتراضية 0 (أوNULLإذا كان مؤشر) بشكل أوتوماتيكي. حيث تعطى هذه القيم البدائية لحقول السجلات إذا تم التصريح عنها خارج نطاق تعريف التوابع(أي إذا كانت متحولات عامة) وذلك إذا لم يتم تمرير قيم بدائية لها بشكل صريح. يمكن إعطاء قيم بدائية لحقول السجلات بشكل أخر وذلك عن طريق الإسناد كما وجدنا سابقاً

2-9 العمليات على السجلات

العمليات التي يمكن تنفيذها على السجلات هي عملية إسناد سجل إلى سجل أخر من نفس النمط، وعملية اخذ عنوان السجل (&) والعملية sizeof لتحديد حجم السجل.لكن عند مقارنة السجلات لن نحصل على القيمة true عند مقارنة سجلين حتى إذا كانا من نفس النمط وذلك بسبب اختلاف لبايتات في المثال البرمجي التالي عرفنا سجل time وصرحنا عن متحولين الأول d مع قيم بدائية والثاني v أسندنا فيما بعد لـ v قيم السجل d طبعنا قيم حقول d ثم v فوجدناها متطابقة وبعدها طبعنا حجم وعنوان السجل d.

```
#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
struct time
{
int h;
int m;
int s;
};
void printu(const time &);
void printsiz (const time &);
void main()
{
   time d={11,29,56};
   time v;
```

```
    v=d;
    cout<<"output  time d:";
    printu (d);
    cout<<"output  time v=d:";
    printu (v);

    cout<<"\n output  size of     and address  d struct :\n";
    printsiz (d);
    system("pause");
    return 0;
}
    void printu (const time &t)
    {
    cout <<t.h<<":"<<t.m<<":"<<t.s;
    }
    void printsiz(const time &t)
    {
    cout<<"size of: "<< sizeof(t)<<"    address: "<<&t<<endl;
    }
```

خرج البرنامج يشبه الشكل:

```
output  time d:  11:29:56

output  time v=d:  11:29:56

output  size of  and address  d struct :
```

10-2السجلات الثابتة

تحتاج بعض السجلات أن تكون ذات طبيعة قابلة للتغير ولا تحتاج البعض منها لذلك. يمكن للمبرمج أن يستخدم الكلمة المفتاح const لتحديد السجلات التي لا يمكن تغييرها حيث تؤدي أية محاولة لفعل ذلك إلى توليد خطأ. فملاً، السطر التالي:

```
Const time noon={12,0,0};
```

يقوم بالتصريح عن السجل noonذات طبيعة ساكنة(ثابت)

11-2الفرق بين المصفوفات والسجلات

1- المصفوفات مسبقة التعريف والسجلات معرفة من قبل المبرمج
2- المصفوفات متجانسة والسجلات غير متجانسة
3- يمكن الوصول إلى مركبات المصفوفة عن طريق الدليل بينما يمكن الوصول إلى حقول السجلات عن طريق اسم السجل نقطة اسم الحقل.
4- تمرر المصفوفات كوسطاء للدوال بالمرجع فقط بينما تمرر السجلات كوسطاء للدوال بالمرجع والقيمة

الصـــفوف

class

3-1 تعريف الصف

يعتبر الصف حجر الأساس في البرمجة كائنيه التوجه في لغة c++ حيث الصف طريقة لترتيب المعطيات (المتحولات)مع الدوال العاملة عليها ضمن بنية واحدة.

تقابل بني المعطيات صف class في لغة c++ بنية السجل struct في لغة c الأم، تساعد الصفوف المبرمج على نمذجة الأغراض opjects التي لها عدة صفوف. ويتم تعريف الصف في لغة c++ (التي تضمن معطيات "متحولات ") تسمى بالمعطيات الأعضاء ، وتوابع تسمى بالتوابع الأعضاء) بالكلمة المفتاح class. يليها اسم للصف وبين {} نضع تعريفات لأسماء متحولات وتوابع ينتهي تعريف الصف بـ(;) بعد تعريف الصف، يمكن استخدام اسمه للتصريح عن الأغراض التي تنتمي إليه.

سوف نتعرف على كيفية بناء الصف من خلال مثال بسيط للصف time

```
class time                              // اسم الصف
{                                       // بداية الصف
 public:                                // قسم الواجهة أو القسم الخارجي
   void input();                        //تابع عضو عام
   void  printu();                      // تابع عضو عام
    void prints();                      //تابع عضو عام
 private:                               // القسم الداخلي أو القسم الخاص
   int h;                               //متحول عضو خاص
   int m;                               //متحول عضو خاص
```

```
int s;                                  متحول عضو خاص//
};                                      نهاية الصف//
```

يبدأ تعريف الصف بالكلمة المفتاح class يليها اسم اختياري للصف (يخضع لقاعدة تعريف الأسماء)، يتم تحديد جسم الصف بواسطة قوسين {} ، ينتهي تعريف الصف دوماً بالفاصلة المنقوطة (;).

ينقسم جسم الصف إلى قسمين أساسيين (يمكن أن يكون أكثر من قسمين كما سنرى فيما بعد) يبدأ كل قسم بكلمات مفتاحيه تسمى المحددات يحدد كل منها مستوى الوصول لهذا القسم:

- المحدد:public (ويعني واجهه أو عام)
- المحدد :private (يعني خاص أو داخلي)

في القسم الأول(:public)توجد نماذج ترويسات الدوال ;printu() & ;prints() ;input()& تسمى هذه الدوال بالدوال الأعضاء العامة، وفي القسم الثاني(:private) صرحنا عن المتحولات h , m , s تسمى بالمعطيات الأعضاء الخاصة.

يفيد التصريح عن المعطيات الأعضاء والدوال الأعضاء بعد المحدد public: في جعل هذه المعطيات والدوال عامة أي يمكن الوصول لها و متاحة للاستخدام من أي نقطة ضمن البرنامج (تسمى أعضاء مصدرة).

أما المعطيات الأعضاء والدوال الأعضاء المصرح عنها بعد المحدد :privateفيمكن الوصول إليها ومتاحة للاستخدام في الدوال الأعضاء المرتبطة بالصف أو الدوال الأصدقاء فقط.

بعد تعريف الصف يمكن استخدامه كنمط للتصريح عن أغراض وذلك كما يلي:

```
time   t;          //كائن( أو غرض) بسيط
time   arayt [];   //كائن( أو غرض) مصفوفة
time   *pt;        //كائن( أو غرض) مؤشر
```

أي أن اسم الصف أصبح كما لو أنه نمط(نوع) مسبق التعريف في c++.وهنا نشير إلى انه يجب التمييز بين الأغراض (مثل t , auayt[] , *pt في المثال السابق)والنوع (time) نفسه.

مثال(1-3):البرنامج الآتي يقوم بطباعة الوقت (الساعة ، الدقيقة، الثانية) باستخدام الدالة;time() المسماة بالدالة البناءة(والتي سوف نتعرف عليها فيما بعد) ودالة للطباعة أسميناها ;show() (وهي تسمية اختيارية).

```
# include <iostream.h>
class   time
{
private:
  int h, m, s;
public:
 void inputtime ()
{
h=10;
m=20;
s=50;
}
void show()
{
cout<<h<<":"<<m<<":"<<s<<"am\n";
}
};  //end class
```

```
 void main()
{
    time   t;
    t.inputtime ();
    t. show();//10:20:50
}// end main
```

يسمى التابع* العضو:time() الذي يحمل نفس اسم الصف بالتابع البناء المرتبط بذلك الصف، يقوم التابع البناء بإعطاء قيم بدائية إلى المعطيات الأعضاء لكل غرض من أغراض الصف، يتم استدعاء التابع البناء آلياً عند إنشاء الغرض.

سوف نرى لاحقاً أنه يمكن أن يوجد في الصف عدة توابع بناءة حيث يتم تحقيق ذلك من خلال التحميل الزائد للتوابع كما يمكن أن لا يكون في الصف أي تابع بناء كما في المثال الأتي .

مثال (2-3):عدل البرنامج السابق ليتم إدخال القيم عن طريق المستخدم

```
    # include <iostream.h>
    class   time
     {
    private:
    int h, m, s;
    public:
     void input();
     void show();
```

```
};
void time :: input()
{
cout<<"input h m s :\n";
cin>>h>>m>>s;
}
```

*ان كلمتي دالة او تابع الواردتين في هذه المطبوعة متطابقتين بالمعنى عن

```
void time :: show()
{
cout<<h<<":"<<m<<":"<<s<<endl;
 }
void main()
```

*ان كلمتي دالة او تابع الواردتين في هذه المطبوعة متطابقتين بالمعنى

```
{
time   t;
t. input();
t. show(); }
```

ملاحظات(1-3):

1-المعطيات الأعضاء h, m, s تقع في الجزء الخاص بالصف :private وهي معطيات خاصة بالصف ولا يمكن الوصول إليها إلى من قبل التوابع الأعضاء للصف. وتهدف هذه العملية إلى حجب المعطيات الخاصة عن الزبائن المستخدمة للصف.

2-لا يمكن إعطاء (إسناد) قيم بدائية للمعطيات الأعضاء الخاصة في مكان التصريح عنها ضمن جسم الصف كما في الحالة التي نهيئ فيها المتحولات في البرنامج، من هنا تبرز أهمية التابع البناء الذي قمنا من خلاله بتهيئة المتحولات.

3-تطابق طريقة التصريح عن التوابع الأعضاء نماذج التصريحات عن التوابع (ترويسات التوابع) التي تعرفنا عليها سابقاً.

4-يمكن تعريف التوابع (كتابة جسم التابع) ضمن الصف كما في المثال (1-3) لكن بفضل القيام بذلك خارج تعريف الصف كما في المثال (2-3).

5-عند تعريف التوابع الأعضاء بعد تعريف الصف تستخدم العملية الثنائية (::) كما في المثال (2-3).

6-بعد التصريح عن الغرض تُستدعى التوابع الأعضاء المتعلقة بالغرض في البرنامج بكتابة اسم الغرض وبعد الـ(.) نكتب اسم التابع إذا كان الغرض من نوع ساكن وتستبدل الـ(.) بـ(<-) إذا كان الغرض من نوع مؤشر.

7-اذا لم نضع اسم المدد صراحة بداية الصف فان المحدد الافتراضي هو :private والذي يمثل الجزء الخاص من الصف.

أمثلة محلولة

مثال (3-3):أكتب برنامج يقوم بحساب مساحة ومحيط الدائرة مع إدخال نصف القطر عن طريق المستخدم؟

```
# include <iostream.h>
float const pi=3.14;
class circle
{
   private:
     float r, area, circ;
  public:
```

```
    void get_r()
{
      cin>>r;
}
void calc_area()
{
      area=pi*r*r;
      cout<<area<<"\n";
}
void calc_circ()
{
     circ=2*pi*r;
     cout<<circ;
}
};
void main()
{
  circle c1;
  c1.get_r();
  c1.calc_area();
  c1.calc_circ();
```

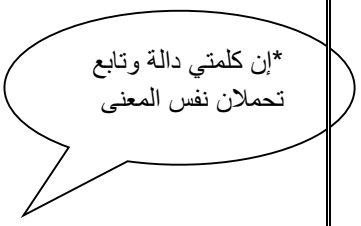
*ان كلمتي دالة او تابع الواردتين في هذه المطبوعة متطابقتين بالمعنى

```
}
```

في هذا المثال تم تعريف التوابع الأعضاء داخل تعريف الصف.

مثال (4-3):أكتب برنامج يقوم بحساب مساحة ومحيط المستطيل مع إدخال الطول والعرض عن طريق المستخدم؟

```
# include <iostream.h>
class mos
{
private:
  float t, a, area, most;
public:
void get_t()
{
    cin>>t;
}
void get_a()
{
   cin>>a;
}
void calc_area()
{
```



```
   area=t*a;
 }
void calc_mos()
{
    most=(t+a)*2;
}
void print()
{
cout<<area<<endl<<most;
}
};
void main()
{
  mos m1;
  m1. get_t();
  m1.get_a();
  m1.calc_area();
  m1.calc_mos();
  m1.print();
}
```

*ان كلمتي دالة او تابع الواردتين في هذه المطبوعة متطابقتين بالمعنى

مثال(5-3):أكتب برنامج يقوم بحساب مضروب عدد مدخل؟

```
# include <iostream.h>
class fact
{
  private:
    int n;
   long f;
 public:
  void  input()
{
cin<<n;
}
  void calc_fact()
{
    f=1;
for(int i=1; i<=n; i++)
f=i*f;
    return 0;
}
void print()
{
```

```
        cout<<n<<"!="<<f<<"\n";
    }
};

   void main()
    {
     fact f1;
    f1.input();
    f1. calc_fact();
    f1. print();
    }
```

2-3 مجال رؤية صف وكيفية الوصول إلى أعضائه

تقع المعطيات الأعضاء والتوابع الأعضاء ضمن مجال رؤية الصف. ويمكن ضمن مجال رؤية الصف الوصول مباشرة إلى أعضائه من قبل كافة التوابع الأعضاء التابعة له وذلك بعد ذكرها بالاسم فقط. أما خارج ذلك المجال، فيمكن الوصول إلى أعضاء صف العامة فقط من خلال اسم الغرض.

تشبه العمليات المستخدمة للوصول إلى أعضاء صف العمليات المستخدمة للوصول إلى حقول السجلات إي باستخدام العملية (.)(وذلك للأغراض الساكنة) أو (<-)(للأغراض من نوع مؤشر).

يستخدم البرنامج التالي صفاً بسيطاً يدعى count يملك هذا الصف معطيات عامة x وتابعاً عاماً print. حيث يبين البرنامج كيفية استخدام العمليتين (.) أو (<-) للوصول إلى أعضاء صف:

```
# include <iostream.h>
class count
{
  public:
    int x;
    void print()
      {
        cout<<x<<endl;
      }
};    //end  class count;
int main()
{
   count r;
   count *pr=&r;
   count &fr=r;
   cout<<"write object";
r. x=1;
r. print();
cout<<"white reference:";
  fr. x =2;
fr.print();
```

```
cout<<"white pointer:";
pr -> x =3;
pr -> print();
}
```

يمتلك الصف count معطيات عامة x وتابع عام print. ويوجد صمن البرنامج ثلاثة متحولات (أغراض) من نمط الصف count وهي r ، fr (عنوان غرض تابع للصف) و pr (مؤشر على عرض تابع للصف).

مثال (6-3):أكتب برنامج يقوم بإدخال عشرة أعداد في مصفوفة ثم يقوم بإيجاد مجموع هذه العناصر ومتوسطها؟

```
# include <iostream.h>
class matrix
{
  private:
    int a[10], sum;
    float avg;
 public:
   void get_a()
   {
     sum=0;
     for(int i=0; i<=9; i++)
```

```
        {
         cin>>a[i];
         sum=sum + a[i];
         }
       }

    void calc_avg()
     {
      avg=sum /10.0;
     }
   void print()
    {
     cout<<"sum="<<sum<<"\n"<<"avg="<<avg;
    }
};
void main()
{
  matrix m1;
   m1.get_a();
   m1.calc_avg();
   m1.print();
```

```
}
```

مثال(7-3):أكتب برنامج يقوم بإدخال عشرة أعداد في مصفوفة ثم يقوم بإيجاد مجموع هذه العناصر ومتوسطها؟ "باستخدام المؤشرات".

```
# include <iostream. h>
class matrix
{
  private:
    int a[10],sum;
    int *p;
    float avg;
  public:
    matrix ()
     {
      p=a;
      sum=0;
      for(int i=0; i<=9; i++)
       {
        cin>>a[i];
       sum=sum + a[i];
        }
    }
```

```
  void calc_avg()
   {
     avg=sum /10;
   }
  void print()
  {
    cout<<sum<<"\n"<<avg;
  }
};
void main()
{
matrix m1;
m1.get_a();
m1.calc_avg();
m1.print();
}
```

3-3 التوابع (أو الدوال) البناءة

ذكرنا سابقاً أن كل صف يمكن أن يحوي على تابع بناء والتابع البناء يحمل نفس اسم الصف. ويمكن أن يكون بوسطاء أو بدون وسطاء. كما انه يمكن أن يوجد في الصف أكثر من تابع بناء وذلك باستخدام

التحميل الزائد للتوابع. يُستخدم التابع البناء لإعطاء قيم بدائية للمعطيات الأعضاء. ويُستدعى بشكل آلي.

3-3-1 التابع البناء بدون وسطاء : في المثالين التاليين توضيح لاستخدام وعمل التوابع البناءة

مثال (3-8):البرنامج الأتي يطبع العبارة hello constructor أربع مرات على شاشة الخروج.

```
# include <iostream.h>
class test
{
public:
test()
  {
    cout<<"hello constructor \n";
  }
};
void main()
{
test t1, t2, t3, t4;
}
```

مثال(3-9):في البرنامج الآتي للصف time تابع بناء

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

```
{
public:
 time();
 void print();
private:
  int h;
  int m;
  int s;
};
time :: time()
{
h =20;
m =37;
s =55;
}
void time :: print()
{
cout<<h<<":"<<m<<":"<<s<<endl;
}
void main()
{
```

```
time t;
t.print();
}
```

لاحظ عدم استدعاء التابع البناء صراحة في المثالين السابقين حيث تم استدعائه آلياً.

2-3-3 التابع البناء مع وسطاء :

عند إنشاء غرض يمكن لأعضائه أن تأخذ قيماً ابتدائية بواسطة التابع البناء المرتبط بالصف. يستخدم التابع البناء مع وسطاء لإعطاء قيم بدائية لكل غرض مشتق من الصف تختلف عن بعضها. حيث أنه عند التصريح عن غرض مرتبط بصف فإنه يمكن تحديد القيم الابتدائية الممكن تمريرها له ضمن قوسين على يمين الغرض وقبل الفاصلة المنقوطة. يوضح المثال الآتي كيفية تمرير الوسطاء ضمنياً إلى التابع البناء.

مثال (10-3):

```
# include <iostream. h>
class time
{
public:
time(int =0,int =0,int =0);
  void print();
private:
  int h, m, s;
};
```

```
time :: time(int h1, int m1, int s1)
{
h =h1;
m =m1;
s =s1;
}
void time :: print()
{
  cout<<h<<":"<<m<<":"<<s<<endl;
}
main()
{
time t1;
time t2(9);
time t3(9, 44);
time t4(9, 44, 55);
t1. print( );                    //      0:0:0
t2. print();                     //      9:0:0
t3. print();                     //      9:44:00
t4. print();                     //      9:44:55
}
```

مثال(3-10):أكتب برنامج يقوم بحساب مضروب عدد مدخل بدالة بناءة مع وسيط؟

```
# include <iostream.h>
class fact
{
  private:
    int n;
   long f;
 public:
   fact(int m)
{
n=m;
}
  void calc_fact()
{
    f=1;
for(int i=1; i<=n; i++)
f=i*f;
    return 0;

}
```

```
    void print()
    {
        cout<<n<<"!="<<f<<"\n";
    }
};

   void main()
    {
     fact f1(5);
    f1. calc_fact();
    f1. print();
    }
```

3-3-3 التحميل الزائد للتوابع البناءة:

التابع البناء يقبل التحميل الزائد والمثال(3-15)من الفقرة (3-5)القادمة يوضح ذلك.

3-4الإسناد الافتراضي للأعضاء

يتم استخدام عملية الإسناد (=) لإسناد غرض إلى غرض آخر من نفس النوع. حيث يتم عادة تنفيذ عملية الإسناد على الغرض باستخدام نسخ الأعضاء (نسخ كل عضو من أعضاء الغرض الأول إلى نفس العضو من الغرض الثاني).

يمكن تمرير الأغراض كوسطاء لتابع ويمكن أيضاً إعادتها كقيم للتوابع. تعتمد تلك العمليات على تمرير أو إعادة نسخه عن الغرض الممرر أو المعاد. المثال الآتي يبين طريقة إسناد غرض إلى آخر:

مثال(3-11):

```
# include <iostream.h>
class date
{
 public:
  date (int =1, int=1, int=1990);
  void print();
private:
  int mouth;
  int day;
  int year;
};
date :: date(int m, int d, int y)
{
  mouth = m;
  day = d;
  year= y;
}
```

```
void date::print()
{
  cout<<day<<"/"<<mouth<<"/"<<year;
}
void main()
{
  date date1(7, 4, 2002);
  date date2;
  cout<<"date1=";
  date1.print();  //   date1= 4/7/2002
  cout<<"\n date2=";
  date2. print();  //  date2=1/1/1990
  date2 =date1;
  cout<<"\n date2= date1= \n";
date2.print();//  date2= date1=4/7/2002
}
```

خرج البرنامج:

```
date1= 4/7/2002
date2= 1/1/1990
```

مثال اخر

```
# include <iostream.h>
class fact
{
  private:
    int n;
   long f;
 public:
   fact(int m)
{
n=m;
}
  void calc_fact()
{
    f=1;
for(int i=1; i<=n; i++)
f=i*f;
    return 0;

}
```

```
    void print()
    {
        cout<<n<<"!="<<f<<"\n";
    }
};
    }
    void print()
    {
        cout<<n<<"!="<<f<<"\n";
    }
};

   void main()
    {
     fact f1(5),f2(3);
    f1. calc_fact();
    f1. print();
    f2. calc_fact();
    f2. print();
    f2=f1;
    f2. calc_fact();
```

```
    f2. print();
    }
```

مثال:

```
#include <iostream.h>
#include <stdlib.h>
class sum
{
 private:
   int a,b,c;
 public:
   sum(int=0,int=0);
   void print();
};
sum::sum(int x,int y)
{
  a=x;b=y;
}
void sum::print()
{
  c=a+b;
```

```
  cout<<a<<"+"<<b<<"="<<c<<endl;
}
int main()
{
 sum s1,s2(6,7),s3(20,70);
 s1.print();
 s2.print();
 s3.print();
    system("PAUSE");
    return 0;
}
```

3-5 الهادمات أو المدمرات

التابع الهادم هو تابع عضو خاص. يحمل نفس اسم الصف مسبوقاً بالحرف (~) واستخدام الحرف (~) يعني أن التابع الهادم له دور متمم للتابع البناء. يقوم التابع الهادم بإنهاء عملية الاحتفاظ بفضاء الذاكرة المخصص للغرض. ليس للتابع الهادم وسطاء ولا يعيد أيه قيمة يكون لكل صف تابع هادم وحيد وهو لا يقبل التحميل الزائد عليه. والتوابع الهادمة تناسب الأغراض المتضمنة قيم ديناميكية كما سنرى فيما بعد. واستدعاء التوابع البناءة المدمرة يتم بشكل ضمني. ويجري عادة استدعاء المدمرات حسب الترتيب العكسي لعمليات استدعاء التوابع البناءة.

يتم استدعاء التوابع الهادمة ضمن مجال الرؤية العام وذلك قبل تنفيذ أيه عملية استدعاء لتابع أخر (قبل حتى main) أو عند استدعاء التابع exit.البرامج الآتية توضح عملية استدعاء التوابع البناءة والمدمرة.

مثال:

```
#include<iostream.h>
class time
{
private:
int h,m,s;
public:
time(int,int,int);
~time();
};
time::time(int x,int y,int z)
{
h=x;
m=y;
s=z;
cout<<"time:"<<h<<':'<<m<<':'<<s<<endl;
}
time::~time()
{
cout<<"~time:"<<h<<':'<<m<<':'<<s<<endl;
}
time t0(0,0,0);
```

```
main()
{
time t1(10,10,10);
time t2(20,20,20);
cout<<"end function main\n";
}
```

```
time:0:0:0
time:10:10:10
time:20:20:20
end function main
```

مثال(12-3):باستخدام دوال البناء والهدم أكتب برنامج يقوم بجمع عددين وطباعة الناتج ؟

```
#include<iostream.h>
class add
{
private:
int a,b,c;
public:
```

```
add();
~add();
void print();
};
add::add()
{
a=10;
b=20;
c=a+b;
}
add::~add()
{
}
void add::print()
{
cout<<"a="<<a<<"\nb="<<b<<"\n c="<<c;
}
void main()
{
add al;
al.print();
```

*إن كلمتي دالة أو تابع الواردتين في هذه المطبوعة متطابقتين بالمعنى

```
}
```

مثال(13-3): باستخدام دوال البناء والهدم أكتب برنامج يقوم بطباعة الوقت (الساعة ، الدقيقة ، الثانية) ؟ ثم عدّل بالبرنامج ليتم إدخال القيم عن طريق المستخدم وذلك بإضافة وسائط الدالة البناء ؟

```
#include<iostream.h>
class time
{
private:
int hour,min,sec;
public:
time();
~time();
void show();
};
time::time()
{
hour=10;
min=20;
sec=30;
}
time::~time()
```

```
{
}
void time::show()
{
cout<<hour<<":"<<min<<":"<<sec<<"\n";
}
void main()
{
time t1,t2,t3;
t1.show();
t2.show();
t3.show();
}
```

مثال(14-3):باستخدام دوال البناء والهدم أكتب برنامج يقوم بطباعة الوقت (الساعة ، الدقيقة ، الثانية) ؟ حيث يتم إدخال القيم عن طريق المستخدم وذلك بإضافة وسائط لدالة البناء ؟

```
#include<iostream.h>
class time
{
private:
int hour,min,sec;
```

```
public:
time(int,int,int);
~time();
void show();
void get_h_m_s()
};
void time::get_h_m_s()
{
cin>>hour>>min>>sec;
}
time::time(int h,int m,int s)
{
hour=h;
min=m;
sec=s;
}
time::time()
{
h=10;
m=10;
s=10;
```

```
}
time::~time()
{
}
void time::show()
{
cout<<hour=<<":"<<min<<":"<<sec<<"\n";
}
void main()
{
int k,n,m;
time t1(k,n,m),t2(k,n,m),t3(k,n,m);
t1.get_h_m_s();
t1.show();
t2.get_h_m_s();
t2.show();
t3.get_h_m_s();
t3.show();
}
```

مثال(15-3): باستخدام دوال البناء والهدم أكتب برنامج يقوم بحساب مساحة ومحيط مستطيل وذلك بواسطة دالتي بناء الأولى بدون وسطاء والأخرى بوسطاء ؟ ثم عدل في البرنامج وذلك بإضافة مؤشر للكائن ؟

```
#include<iostream.h>
class rect
{
private:
int h,w;
public:
rect ();
rect(int,int);
~ rect ();
void prect ();
};
rect:: rect ()
{
h=5;
w=7;
}
rect:: rect(int a,int b)
{
h=a;
w=b;
```

```
}
rect::~ rect ()
{
}
void rect:: prect ()
{
cout<<" rect ="<<(w*h)<<"\n";
}
main()
{
rect r1,r2(3,8);
r1. prect ();
r2. prect ();
}
```

3-6 الدوال set,get (أو الدوال الخطية)

توجد عادة في الصفوف عدد من التوابع الأعضاء العامة التي من خلالها يستطيع الزبون تعديل قيم المعطيات الخاصة ليس من الضروري أن نسمي التوابع set,getولكن جرت العادة استخدام هذين الاسمين.

سنقوم في البرنامج الآتي بإضافة التابعين set,getمن أجل المتحولات الأعضاء. تقوم التوابع set بالمراقبة المباشرة للقيم المعطاة للمعطيات الأعضاء ويقوم التابع get بإعادة قيمة المعطيات الأعضاء التي نريد الحصول عليها.

مثال(3-16):باستخدام دوال البناء والهدم أكتب برنامج يقوم بحساب مساحة مستطيل باستخدام دالة خطية ؟ وذلك بإضافة مؤشر للكائن ؟

```
#include<iostream.h>
class rect
{
private:
int h,w;
public:
void set_value(int,int);
int area();
};
void rect::set_value(int u,int v)
{
h=u;
w=v;
}
int rect::area()
{
return(h*w);
}
void main()
```

```
{
rect a,*b,*c;
rect *d=new rect[2];
c=&a;
b=new rect;
a.set_value(3,5);
b->set_value)8,10);
d[0]->set_value(3,7);
d[1].set_value(4,8);
cout<<c->area()<<"\n";
cout<<b->area()<<"\n";
cout<<d[0].area()<<"\n";
cout<<(*(d+1)).area()<<"\n";
cout<<d[1].area()<<"\n";
}
```

مثال(17-3):باستخدام الدوال الخطية set,get أكتب برنامج يقوم بحساب مساحة ومحيط الدائرة مع إدخال نصف القطر عن طريق المستخدم ؟

```
#include <iostream.h>
class circle
{
private:
```

```
float r, area,circ;
public:
void get_r();
void set_r();
void calc_area();
void calc_circ();
void print();
};
void circle::get_r()
{
cin>>r;
}
void circle::set_r()
{
if(r<0) r=0;
}

void circle::calc_area()
{
area=3.14*r*r;
}
```

```
void circle::calc_circ()
{
circ=2*3.14*r;
}
void circle::print()
{
cout<<r<<"\n"<<area<<"\n"<<circ;
}
void main()
{
circle c1;
c1.get_r();
c1.set_r();
c1.calc_area();
c1.calc_circ();
c1.print();
}
```

في البرنامج تم إدخال القيم للأعضاء الخاصة من خلال التابعget_r()وتم مراقبة هذه القيم بواسطة التابع

.set_r()

مثال(18-3):باستخدام الدوال الخطية أكتب برنامج يقوم بالعمليات الحسابية (الجمع والطرح والضرب والقسمة) لعددين مدخلين ثم أعد كتابة البرنامج باستخدام المؤشرات ومرة أخرى بالمراجع ؟

```
#include <iostream.h>
class opr
{
private:
int x,y;
public:
int add();
int subt();
int mult();
int dive();
void print();
void get_xy(int,int);
void set_y();

};
void main()
{
opr op;
int a,b;
```

```
op.get_xy(a,b);
op.add();
op.subt();
op.mult();
op.set_y();
op.dive();
op.print();
}
void opr::get_xy(int a,int b)
{
cin>>a>>b;
x=a;
y=b;
}
void opr::set_y()
{
if(y==0) y=1;
}

int opr::add()
{
```

```
return(x+y);
}
int opr::subt()
{
return(x-y);
}

int opr::mult()
{

return(x*y);
}
int opr::dive()
{

return(x/y);
}
void opr::print()
{
cout<<add()<<"\n"
```

```
<<subt()<<"\n"
<<mult()<<"\n"
<<dive();
}
```

في هذا المثال تمت عملية المراقبة على المتحول y حتى لا تتم القسمة على 0.

مثال(19-3): باستخدام الدوال الخطية أكتب برنامج يقوم بالعمليات الحسابية (الجمع والطرح والضرب والقسمة) لعددين مدخلين وذلك باستخدام المؤشرات ؟

```
#include <iostream.h>
class opr
{
private:
int x,y;
public:
int add();
int subt();
int mult();
float dive();
```

```
void print();
void get_xy(int *,int *);
void set_y();

};
void main()
{
int *a,*b;
opr op;
op.get_xy(a, b);
op.add();
op.subt();
op.mult();
op.set_y();
op.dive();
op.print();
}
void opr::get_xy(int *a,int *b)
{
cin>>*a>>*b;
x=*a;
```

```
y=*b;
}
void opr::set_y()
{
if(*y==0)  *y=1 ;
}

int opr::add()
{
return(x+y);
}
int opr::subt()
{
Return(x-y);
}
int opr::mult()
{
Return(x*y);
}
int opr::dive()
{
```

```
return(x/y));
}
void opr::print()
{
cout<<add()<<"\n"
<<subt()<<"\n"
<<mult()<<"\n"
<<dive()
}
```

مثال(20-3):باستخدام الدوال الخطية أكتب برنامج يقوم بالعمليات الحسابية (الجمع والطرح والضرب والقسمة) لعددين مدخلين وذلك باستخدام المراجع ؟

```
#include <iostraem.h>
class opr
{
private:
int x,y;
public:
int add();
int subt();
int mult();
float dive();
```

```
void print();
void get_xy(int&,int&);
void opr::set_y();
};
void main()
{
int a,b;
opr op;
op.get_xy(&a,&b);
op.add();
op.subt();
op.mult();
op.set_y();
op.dive();
op.print();
}
void opr::get_xy(int&a,int&b)
{
cin>>a>>b;
x=a;
y=b;
```

```
}
void opr::set_y()
{
if(y==0) y=1;
}
int opr::add()
{
return(x+y);
}
int opr::subt()
{
return(x-y);
}
int opr::mult()
{
return(x*y);
}
int opr::dive()
{
return(x/y);
}
```

```
void opr::print()
{
cout<<add()<<"\n"
<<subt()<<"\n"
<<mult()<<"\n"
<<dive()
}
```

3-7 الفصل بين الواجهات والنصوص البرمجية

تعتبر عملية الفصل بين الواجهات والنصوص البرمجية أحد أهم المبادئ الأساسية لصياغة البرامج الجيدة لأن ذلك يسهل عملية تعديل البرامج وتطويرها حيث أن أي تعديل يمس النصوص البرمجية لتوابع الصف لا يؤثر عليها طالما أنه لا يوجد أي تعديل يخص واجهات التعامل مع هذه التوابع (أي يمكن تطوير طريقة عمل الصف مع الحفاظ على واجهة التعامل معه ثابتة). وفق هذا المنهج نقوم بتجزئة البرامج ضمن عدة ملفات (برنامجين) وهما :

الأول : (الملف الأول) نضع فيه تعريف الصف و تعريفات التوابع ضمن ملف رأسي

الثاني :وهو البرنامج التنفيذي يتضمن استخدام (استدعاء) للصف المعرف سابقاً

المثال الآتي يتضمن تقسيم البرنامج في المثال (10-3) إلى ملفين :

مثال(3-21):

i-برنامج الملف الرأسي المتضمن الصف time

```
// fig 1 : time 1.h
# ifndef  time1 _h
```

```
#define time 1_h
class  time {
public:
time( );
void settime(int,int,int);
void printv( ) ;
void prints( ) ;
private:
int  h,m,s;
};
time::time( )
{
h=m=s=0;
}
void  time::settime(int h,int m,int s)
{
h=(h>=0&&h<24)? h:0;
m=(m>=0&&m<60)? m:0;
s=(s>=0&&s<60)? s:0;
}
```

```
void  time::printv( )
{
Cout<<h<<":"<<m<<":"<<s;
}
void  time::prints( )
{
cout<<((h==0||h==12)?12:h%12)<<":"<<m<<":"<<s<<h<<12?"a
m":  "pm";
}
#endif
```

ii-الملف التنفيذي (البرنامج الذي يستدعي الصف) للصف time

```
// fig 2 : fig.cpp
#include<iostream.h>
#include   "time1.h"
void main ( )
{
```

```
 time t;
t.printv( );
t.prints( );
t.settime(23,27,6);
t. printv( );
t.prints( );
return 0;
}
```

إن خرج البرنامج السابق هو:

```
00:00:00
12:00:00 am
13:27:06
```

بشكل عام :

لبناء برنامج في لغة c++ يجب وضع كافة تعريفات الصفوف مع تعريفات التوابع ضمن ملفات رأسية تحمل أسماً معبراً عن الهدف منها. يتم كتابة الملفات الرأسية بحيث يكون ملف لكل صف يتم إدراج الملفات الرأسية لكل صف باستخدام التوجيه #include بعدها يتم ترجمة كل ملف من الملفات الرأسية وربطها مع الملف الذي يتضمن البرنامج الرئيسي main().

البرنامج الأول يتضمن الملف الرأسي time1.h الذي نجد فيه تعريفاً للصف timeو النصوص البرمجية للتوابع الأعضاء المرتبطة بالصف time أما البرنامج الثاني نجد فيه البرنامج الرئيسي ()main.

لاحظ الملف الثاني يتضمن الملف الرأسي time1.h حيث تم إدراجه باستخدام التوجيه# include ووضع اسم الملف الرأسي بين شرطتين مزدوجتين (" ") بدل من إشارتي (<>) حيث يجري عادة وضع الملفات الرأسية المعرفة من قبل المبرمج ضمن نفس الدليل المتضمن للبرنامج وبالتالي عندما تصادف مرحلة ما قبل الترجمة توجيها لتضمين ملف رأسي # include فأنها تبحث عنه ضمن نفس الدليل فإذا لم تجده فأنها تقوم بالبحث عنه ضمن الملفات الخاصة بمكتبة c++ المعيارية. أما الملفات الرأسية الموضوعة ضمن (<>) فإنها تفترض أنه في مكتبةc++ المعيارية وتبحث عنها هناك.

لاحظ أن التصريح عن الصف time موجود ضمن توجيهات مرحلة ما قبل الترجمة بين التوجيهان

#define time_h و #endif.

```
# ifndef time_h
#define time_h
- - - - - - - - - - - -
- - - - - - - - - - - -
#endif
```

عند بناء وتطوير البرامج الكبيرة يساعد التوجيهان # define و #ifndef على عدم إدراج النص البرمجي المحصور بينهما إذا كان الاسم time.h معرف من قبل فسيتم إدراج محتوياته أي أن التوجيهان السابقان يفيدان في إدراج محتويات ملف مرة واحدة فقط حتى لا نقع في خطأ إدراج الملفات الرأسية أكثر من مرة والذي يحدث عادة عند تطوير البرامج الكبيرة.

الملفان السابقان مثال(3-12) لهما خرج يشبه خرج المثال(3-10).

8-3 استخدام الوسطاء الافتراضية مع الدوال البناءة

وجدنا سابقاً أنه يمكن كتابة التابع البناء بدون وسطاء أو مع وسطاء في الصف time وذلك لإعطاء قيم بدائية للمتحولات الأعضاءs,m,h.سوف نعيد كتابة المثال (10-3) بتوزيع الملف السابق إلى ملفين وبحيث يكون فيه استدعاء لتابع أخرsettime(int,int,int)والذي لا يعتبر نسخة جديدة من التابع البناء، حيث تم كتابة التابع البناء بحيث يتضمن نفس تعليمات التابع settime()(تعديل التابع settime() سوف يؤدي بشكل أوتوماتيكي إلى تعديل التابع time():

مثال(3-22):

1- الملف الرأسي

```
// fig6.1:time2.h
#ifndef time2_h
#define time2_h
class time{
public:
time ( int=0,int=0,int=0);
void settime(int,int,int);
void printu();
void prints();
private:
int  h,s,m;
};
time::time(int hr,int min,int sec )
{
```

```
 settime(hr,min,sec);
}
void  time::settime(int h,int m,int s)
{
h=(h>=0&&h<24)?h:0;
m=(m>=0&&m<60)?m:0;
s=(s>=0&&s<60)?s:0;
}
void time::printu()
{
cout<<h<<":"<<m<<":"<<s;
}
void  time::prints( ) ;
{
cout<<((h==0||h==12)?12:h%12)<<":"<<m<<":"<<s<<(h<12)?"a
m":  "pm";
}
#endif
```

2- الملف التنفيذي

```
// fig 2 : fig.ccp
```

```
#include"iostream.h"
#include   "time1.h"
void main ( )
{
 time t;
t.printv( );
t.prints( );
t.settime(23,27,6);
t. printv( );
t.prints( );
return 0;
}
```

مثال(3-23):اكتب برنامج للصف creat يوضح كيفية استدعاء تابع الهدم بشكل عكسي لاستدعاء التابع البناء.

```
#ifndef create_h
#define create_h
class creat{
public:
creat(int ,char*);
~creat();
private:
int obj;
```

```
char*mes;
};
creat::creat (int b,char *p)
{
 obj=b;
mes=p;
cout <<"object"<<obj<<"comstuector runs"<<mes<<endl;
}
creat:: ~creat()
{
cout<<"object"<<obj<<"destractor runs"<<mes<<endl;
}
#endif
```

البرنامج الرئيسي

```
//fig6.17:fig6-17.cpp
#include <iostraem.h>
#include"create.h"
creat firstc(1,"global befors main");
main( )
{
```

```
cout<<"main function\n";
creat second(2,"(local in main)");
creat third(3,"(local in main)");
cout<<"end main function"<<endl;
}
```

خرج البرنامج السابق من الشكل

```
object 1 constractor runs global bafors main
main function
object 2 constractor runs local in main
object 3 constractor runs local in main
end main function
object 3 constractor runs local in main
object 2 constractor runs local in main
object 1 constractor runs global bafors main
```

9-3التوابع الأصدقاء والصفوف الأصدقاء

التابع الصديق ليس بتابع عضو من أعضاء الصف لكن يمكنه الوصول إلى أعضاء الصف الخاصة .يتم التصريح عن التابع الصديق داخل الصف وتعريفه خارج مجال رؤية الصف ويكون له الحق في الوصول إلى المعطيات الخاصة. للتصريح عن تابع على أنه صديق لصف يجب إضافة الكلمة المفتاح friend قبل اسم التابع عند التصريح عنه ضمن الصف فقط.

مثال(3-24):يبين البرنامج الآتي كيفية التصريح واستخدام التوابع الصديقة من أجل إعطاء قيمة المتحول المعطيات الخاصة x.

```
//fig7.11:fig7//.cpp
#include <iostream.h>
class count
{
friend void setx(count &,int);
public:
count()
{
x=0;
}
void  print() const
{
cout<<x<<endl;
}
private:
int x;
};
void setx (count &c,int val)
{
c.x=val;
}
```

```
int main()
{
count t,q;
t.print();
setx(t,8);
t.print();
setx(q,20);
q.print();

}
```

لاحظ أن التصريح عن التابع الصديق setx يظهر أولاً ضمن التصريح عن الصف قبل التصريح عن أعضاء الجزء public وهو تابع منفصل وليس تابع عضو ضمن الصف count. لذلك عندما يجري استدعاء التابع setx من أجل الغرض t يجري تمرير الغرض t إلى التابع setx على شكل وسيط بدلاً من صيغة لاستدعاء t.setx(8).

ملاحظة : لا تستطيع التوابع غير الصديقة أو غير الأعضاء الوصول إلى أعضاء الخاصة بصف.

تمارين محلولة

تمرين(1-3):باستخدام دالة البناء والدوال الصديقة أكتب برنامج يقوم بقراءة متغير ثم زيادته بمقدار خمسة عن طريق الدالة الصديقة؟

```
#include <iostraem.h>
```

```
class xx
{
private:
int x;
public:
xx()
{
x=0;
}
void set(int y)
{
x=y;
}
int get()
{
return x;
}
friend void pluss(xx *);
};
void pluss(xx*xx1)
{
```

```
xx1->x+=5;
}
main()
{
xx x1,x2;
x1.set(50);
pluss(&x1);
cout<<x1.get();
}
```

تمرين(2-3): باستخدام دوال البناء والدوال الصديقة أكتب برنامج يقوم بطباعة عددين ثم يقارن بينهما وطباعة العبارة "true"t أو العبارة "false"f؟

```
#include <iostraem.h>
class maxv
{
private:
int z;
public:
maxv (int z1)
{
z=z1;
}
```

```
void print() const
{
cout<<z<<"\n";
}
friend char compare(maxv, maxv);
};
char compare(maxv z1 maxv z2)
{
return(z1.z>z2.z)?'t':'f';
}
main()
{
maxv z1 (10),z2 (20);
z1.print();
z2.print();
cout<<compare(z1,z2);
}
```

تمرين(3-3):كتب برنامج يقوم بمقارنة بين صنفي الريال والدولار باستخدام الأصناف الصديقة ؟

```
#include <iostraem.h>
class riyal
```

```
{
private:
float r;
public:
riyal(float rr=0.0)
{
r=rr;
}
void set(float r1)
{
r=r1
}
float get()const
{
return r;
}
friend bool compare(riyal&,dollar&);
};
class dollar
{
float d;
```

```
public:
dollar(float dd=0.0)
{
d=dd;
void set(float d1)
{
d=d1;
}
float get()const
{
return d;
}
friend bool compare(riyal&,dollar&);
};
bool compare(riyal&r,dollar&d);
{
return(d.d>=3.75*r.r)?"true":"false";
}
main()
{
riyal ry;
```

```
dollar dr;
ry.set(50);
dr.set(30);
if(compare(ry,dr))
cout<<dr.get()<<">="<<ry.get();
else
cout<<<dr.get()<<"<"<<ry.get();
else
cout<<dr.get()<<"<"<<ry.get();
}
```

10-3توابع الوصول وتوابع الأدوات

بالإضافة إلى التوابع العامة التي توضع في واجهة الصف والتي يمكن استخدامها والوصول لها في أي مكان من البرنامج، توجد توابع يمكن استخدامها من قبل التوابع الأخرى التابعة للصف فقط تسمى بالتوابع الخاصة أو بالتوابع الأدوات وتكتب ضمن القسم الخاص بالصف.

مثال(3-25):

يوضح البرنامج الآتي (المؤلف من ملفين) مفهوم التابع الأداة الذي يكتب في القسم الخاص بالصف (ليس في واجهة الصف) ومخصص لأعمال الدعم والمساندة لتنفيذ المهام المناطة بالتوابع الأعضاء العامة المتصلة بالصف ، وهي غير مخصصة للاستخدام من قبل زبائن الصف.

```
//fig6.9:salosp.h
#ifndef salosp _h
#define salosp _h
```

```
class salosperson {
public:
salesperson();
void getsales();
void setsales(int,double);
void writesales();
private:
double totsales();
double sal[12];
};
salesperson:: salesperson()
{
for (int i=0;i<12;i++)
sal [i]=0.0;
}
void salesperson::getsales()
{
cout<<"enter sales amount for month"<<i<<":";
cin>>f;
setsales(i,f);
}
```

```
}
void salesperson:: salesperson(int month,double amout)
{
y(mouth>=1&&month<=12&&amount>0)
sales[month-1]=amounh;
else
cout<<inrlid"<<endl;
}
void salesperson::witesales()
{
cout< salespecision(2)<<fixed<<"\n the total annual sales
wee:$"<<totsales()<<endl;
}
double t=0.0;
for (int i=0;i<12;i++)
t+=sales[i];
return t;
}
#endif
```

يمتلك الصف salesperson مصفوفة مؤلفة من 12 خانة لعمليات البيع الشهرية تأخذ هذه الخانات القيمة الافتراضية صفر بواسطة التابع البناء salesperson بتعديل هذه القيم ويقوم التابع العضو writesales بطباعة مجمل عمليات البيع خلال أشهر السنة يساهم التابع الأداة totsales بحساب

مجموع عمليات البيع. البرنامج الآتي يقوم بإدراج مجموعة من الاستدعاءات لتوابع الأعضاء ضمن التابع main. لقد جرى إخفاء شكل المصفوفة sales بشكل كامل والتوابع الأعضاء باستخدام الصف salesperson.

```
//fig 6.11:fig.cpp
#include "salesp.h"
int main()
salesperson s;
s.getsales()
s.writesales();
return 0;
}
```

11–3الأغراض الساكنة والتوابع الأعضاء الساكنة

1–11–3 الأغراض الساكنة :

تحتاج بعض الأغراض أن تكون ذات طبيعة غير قابلة للتغير ولا يحتاج البعض الآخر لذلك. يمكن عمل ذلك باستخدام الكلمة المفتاح const لتحديد غرض ساكن.

على سبيل المثال الآتي

```
const time cons(12,0,0);
```

هو تصريح عن غرض ساكن const مشتق من الصف time ويأخذ من الغرض القيمة 12 ظهراً لقيمة له.وتفيد هذه العملية في تنفيذ المترجمات لعملها بشكل أسرع من استخدام الأغراض الغير ساكنة (المتغيرة) الأخرى.

أن التصريح عن أغراض ساكنة تمنع من استدعاء أي تابع عضو متعلق بالغرض الساكن. وقد نحتاج إلى ذلك في بعض الأحيان باستدعاء التابع get لمعرفة المعطيات المتعلقة بالغرض دون الحاجة إلى تغيرها. لتحقيق ذلك ، يمكن التصريح عن توابع أعضاء ساكنة بدل التصريح عن أغراض ساكنة. ولأعضاء الساكنة لا يمكنها أن تقوم بتعديل الغرض.

2-11-3 المعطيات الأعضاء الساكنة والتوابع الأعضاء الساكنة:

يمكن تعريف تابع عضو ساكن أثناء التصريح عنه وأثناء تعريفه أيضاً باستخدام الكلمة المفتاح constأو static والتي ترد بعد قائمة الوسطاء وقبل قوس البداية (عند تعريف التابع) ({}) ولا يمكن التصريح عن التوابع البناءة والمدمرة على أنها ساكنة.

البرنامج الآتي يقوم بتعريف صف myconst فيه توابع أعضاء get_x() و get_x() و get_y() ساكنة حيث تم إدراج الكلمة const في أثناء التصريح عند ترويسة الصف في الملف الرأسي وأيضاً أثناء تعريف توابع الصف وأيضاً في التابع الرئيسي main() تم التصريح عن غرضين الأول c1 غير الساكن والغرض c1 ساكن.

مثال(3-26):

```
#include <iostraem.h>
class mycons
```

```
{
private:
int x;
const int y;
public:
myconst();
void set_x();
void get_x()const;
void set_y(int)const;
void get_y()const;
};
myconst::myconst():y(20)
{
x=10;
}
void myconst::set_x()
{
x=x+5;
//y=y+5;
}
void myconst::get_x()const
```

```
{
cout<<x<<"\t";
}
void myconst::set_y()(int y1)const
{
//x=x+5;
y1=y+5;
cout<<"y1="<<y1<<"\n";
}
void myconst::get_y()const
{
cout<<y<<"\n";
}
main()
{
int k;
myconst c1;
const myconst c2;
c1.set_x();
c1.get_x();
c1.set_y(k);
```

```
c1.get_y();
//c2.set_x();
c2.get_x();
c2.set_y(k);
c2.get_y();
}
```

و يمكن التصريح عن متحولات وتوابع أعضاء ساكنة من الصف باستخدام الكلمة المفتاحstatic

مثال(3-27):المثال الآتي يصرح فيه عن متحول عضو ساكن

```
#include <iostraem.h>
class xx
{
private:
static int x=2;
public:
static void print()
{
cout<<x<<"\n";
}
xx()
{
x++;
```

```
}
};
void main()
{
xx x1,x2,x3;
x1.print();
x2.print();
x3.print();
//xx::print();
}
```

اكتب ونفذ البرنامج السابق وفسر خرجه