سي++

إحياء

نسخة إلكترونية - الإصدار :1

مدخل إلى الخوارزميات و البرمجة

فهرس

كيف أستخدم الكتاب :

السلام عليكم و رحمة الله تعالى و بركاته إن الحمد لله نحمده و نستعين به و نعوذ به من شرور أنفسنا و من سيئات أعمالنا من يهده الله فهو المهتد و من يضلل فلن تجد له وليا مرشدا ، و هذا الكتاب قد وضعناه بين أيديكم و أوقفناه في سبيل الله تعالى لأمتنا المسلمة لا نبغ به جزاءً و لا شكوراً إلا رضاً منه جل وعلى ، يتألف الكتاب من جزئين:

النظري:
فيه الأساس النظري للبرمجة و بعض التمارين التطبيقية المباشرة .

العملي:
و فيه مجموعة من التمارين الإضافية على المواضيع المطروحة في الجزء النظري .
و يحوي على 38 سؤال في تلك المواضيع

مع تحيات فريق إحياء للترجمة و التأليف .

ملاحظة هامة جداً
المواد التي تكتب و تحضّر من قبل الفريق هي محاولة لإصلاح الواقع العربي الإسلامي من حيث التأليف و النشر على مستوى الإنترنيت لذلك هذه المواد ليس للتدريس و لاتصلح أن تكون مرجع لاحتمال ورود الأخطاء و ذلك مع حرصنا على عدم ورودها ... لذلك كل مستخدم للمواد المنتجة من قبل الفريق سيتخدمها على مسؤليته الشخصية

مدخل إلى الخوارزميات و برمجة

مقدمة:

بسم الله الرحمن الرحيم و الصلاة و السلام على محمد خاتم النبيين ، هذا الكتاب يشكل أساس بسيطاً يمكنك من خلاله الدخول في عالم البرمجة و يوفر لك قاعدة لفهم طريقة عمل البرامج من خلال بعض الأمثلة التطبيقية .

الخوارزميات

من أين جاءت هذه التسمية (الخوارزميات)

الخوارزميات algorithm مفهوم رياضي قديم يعود إلى مطلع القرن التاسع الميلادي و يعود الفضل في إيجاد هذا المفهوم إلى العالم موسى الخوارزمي لكن هذا المصطلح تطوّر مع الزمن ليرتبط مؤخرا ارتباطاً وثيقاً لبرمجة الحواسيب .

تصريف الخوارزمية :

هي مجموعة متسلسلة من الخطوات المحددة و المنتهية و اللازمة لإنجاز عمل ما أو لحل مسألة ما و الحصول على نتيجة صحيحة .

أنواع الخوارزميات:

-خوارزميات غير حسابية

و هي الخوارزميات التي لا نستخدم فيها العمليات الحسابية مثال عليها خوارزمية التدقيق الإملائي .

-خوارزميات حسابية

مثال :خوارزمية حساب القيمة $y=(3x+3)/(3x-4)$
1/ معرفة قيمة الـ x (المدخل)
2/ حساب قيمة البسط ($u=2x+3$)
3/ حساب قيمة المقام ($v=3x-4$)
4/ حساب قيمة قسمة البسط على المقام (u/v)
5/ طباعةالنتيجة (خرج)

طرق كتابة الخوارزميات:

يمكن صياغة الخوارزميات بطرائق تتفاوت فيما بينها من حيث دقة التعبير و سهولة الفهم و أهم الطرائق المستخدمة لكتابة الخوارزميات:

- الطريقة اللغوية
- الطريقة الترميزية
- الطريقة البيانية(المخططات التدفقية flow chart)

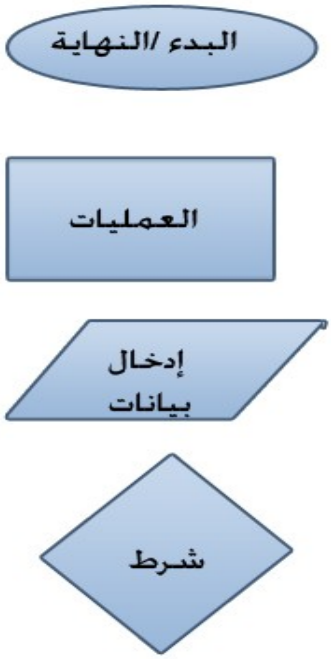
الطريقة الترميزية (طريقة شبه البرنامج)

تعتمد على قواعد محددة يمكن أن تكون مستنتجة من المفاهيم الرياضية و تستخدم في هذه الطريقة الرموز الرياضية بدلاً من اللغة المحكية .
مثال على خوارزمية مكتوبة بهذه الطريقة :
خوارزمية حساب مساحة مستطيل

بفرض الطول L و العرض W

1. input L,W
2. s=L*w
3. print s(إظهار الناتج)
4. end

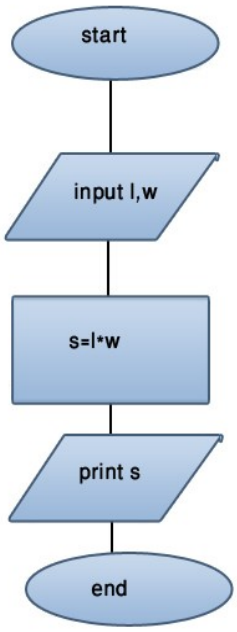
و في ما يلي شرح لهذه الأشكال :



أنواع المخططات التدفقية :

1- المخطط التدفقي التتابعي

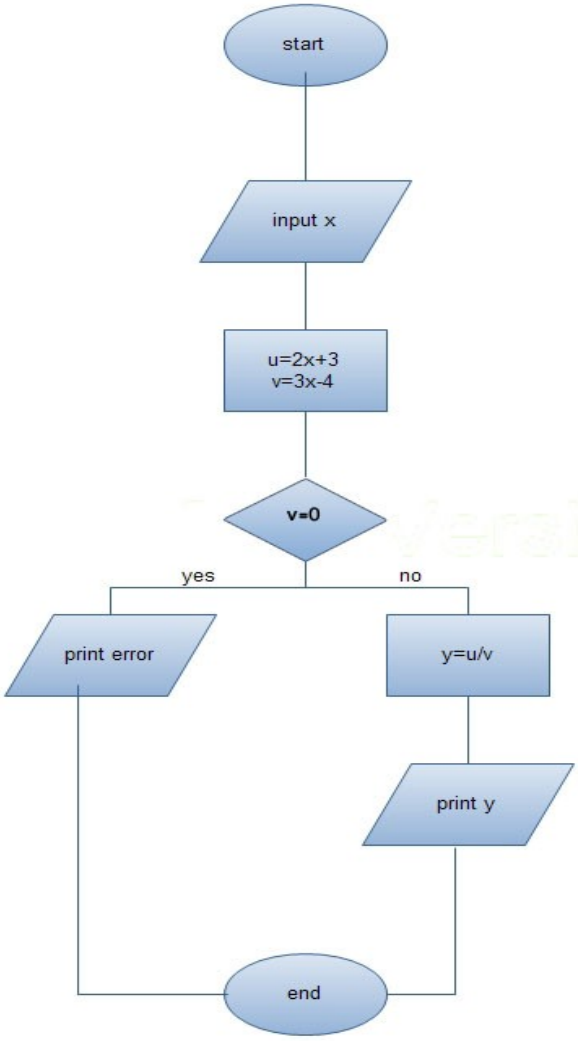
يستخدم هذا المخطط في المسائل المتفرّقة التي يقتضي حلّها تتالياً محدداً للخطوات التي تؤدي إلى النتيجة دون الحاجة إلى تغيير سياق التنفيذ و دون تكرار أو تجاهل لأي خطوة.



مثال على ذلك :
المخطط التدفقي اللازم لحساب مساحة مستطيل

2- المخطط التدفقي التفرعي:

يستخدم هذا النوع في المسائل التي تخضع لشروط تحدد التتالي المناسب للخطوات المطلوب تنفيذها حيث تحقيق الشرط أو عدم تحققه يؤدي إلى الاختيار بين طريقتين أو عدّة طرق .
مثال :
خوارزمية حساب قيمة الكسر
$y=(2x+3)/(3x-4)$

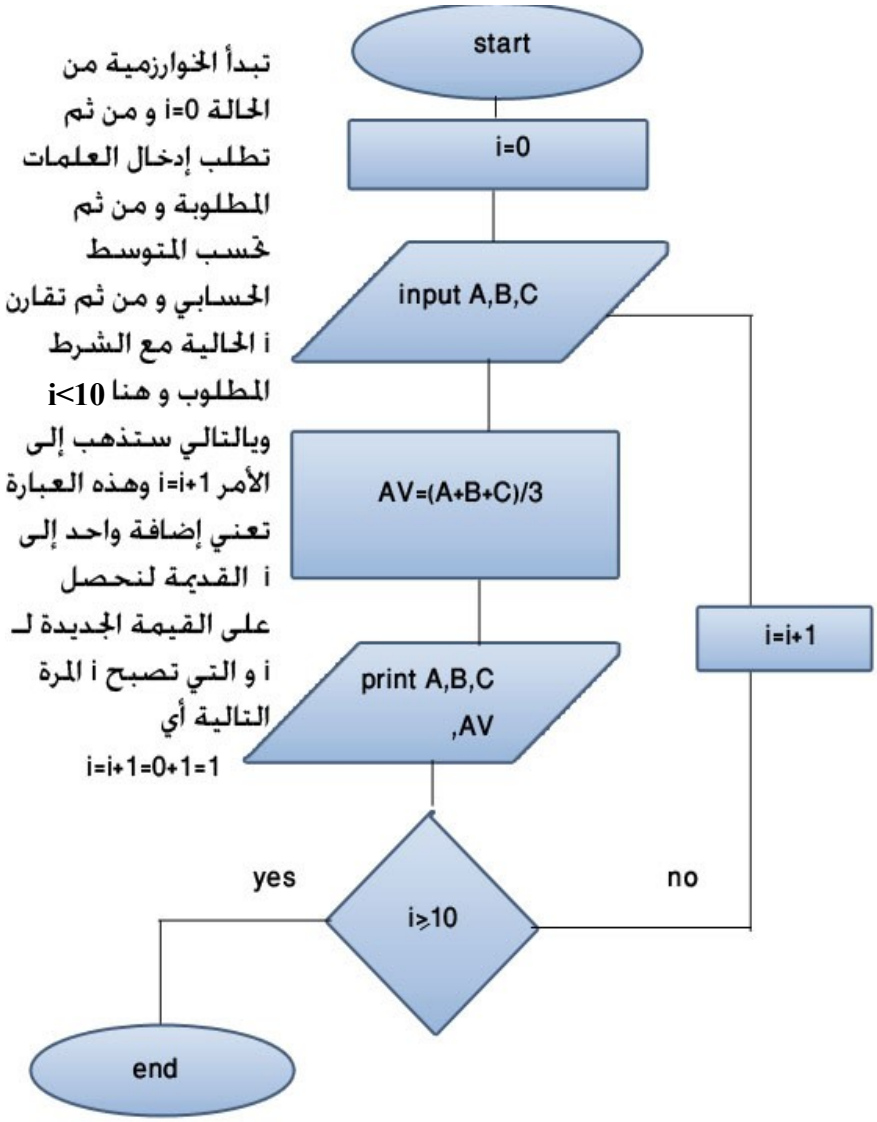


المخطط التدفقي الحلقي :

يستخدم في حل المسائل التي يتضمن حلها تكرار خطوة واحدة أو عدة خطوات مرة واحد أو أكثر حيث يحدد الشرط الذي يقرر عدد مرات التكرار .
مثال :

لدينا 10 طلاب و نرغب في إدخال درجات الطلاب في ثلاث مقررات و من ثم حساب و طباعة معدلات الطلاب مع الدرجات لكل طالب .

بفرض أن المواد هي A,B,C و المتوسط AV و i هو رقم الحالة



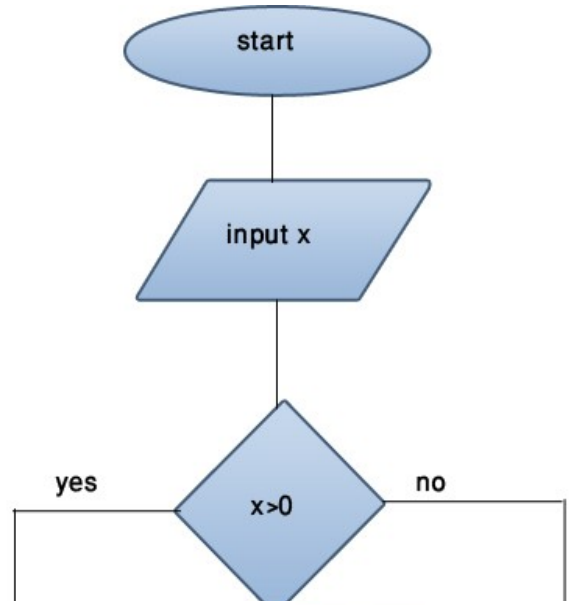
تمارين محلولة على المخططات التدفقية :

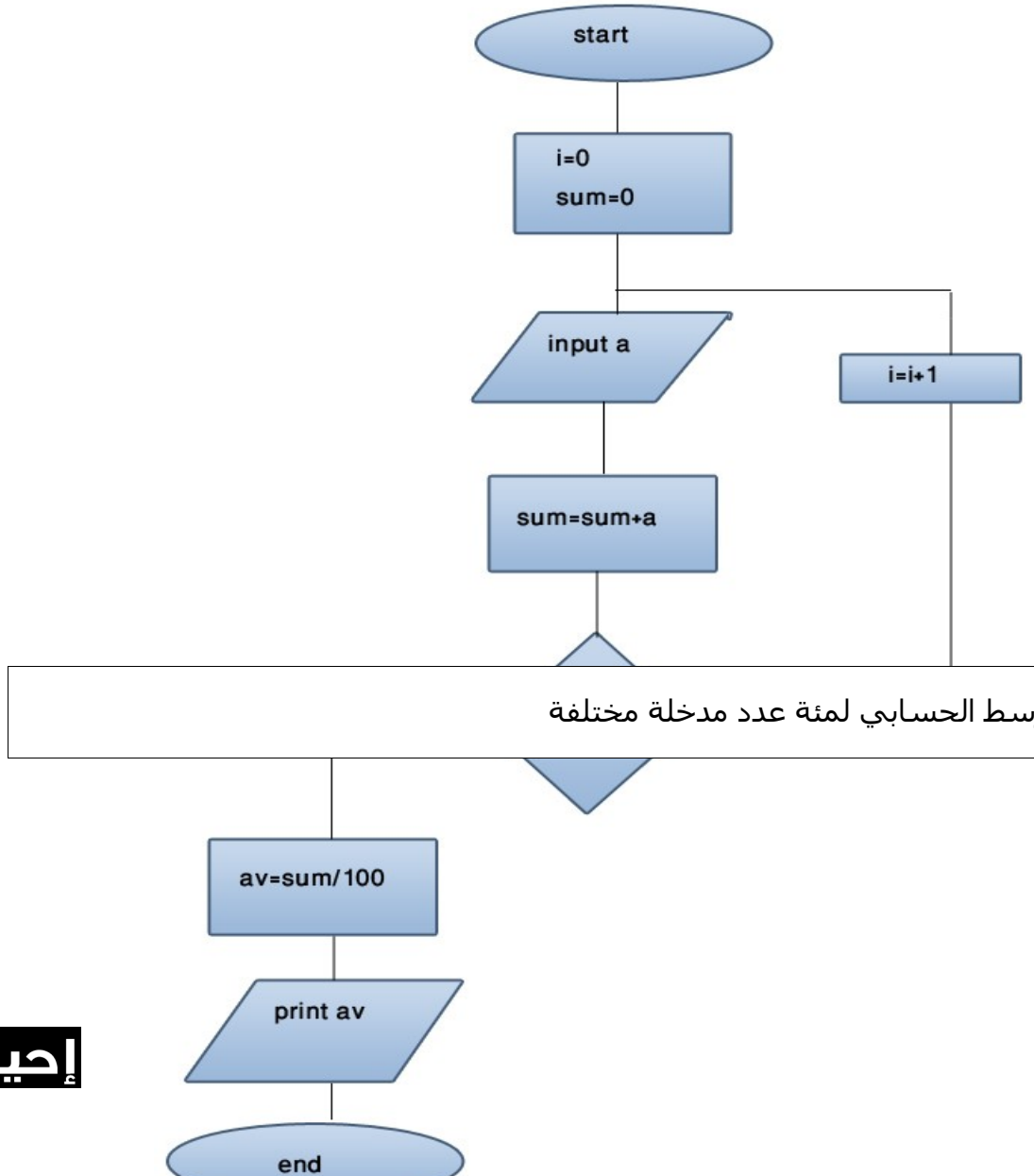
التمرين الأول :

ضع المخطط التدفقي اللازم لحساب و طباعة قيمة الدالة

$$S=\begin{cases}\sqrt{x}+d \ if \ x>0\\ x-a \ if \ x\leqslant 0\end{cases}$$

الحل:
من المفضّل أن تحاول بنفسك أولاً .





التمرين الـ

بفرض أن

و نلاحظ هنا أننا فرضنا قيمة ابتدائية للـ sum وهي الصفر التعليل :

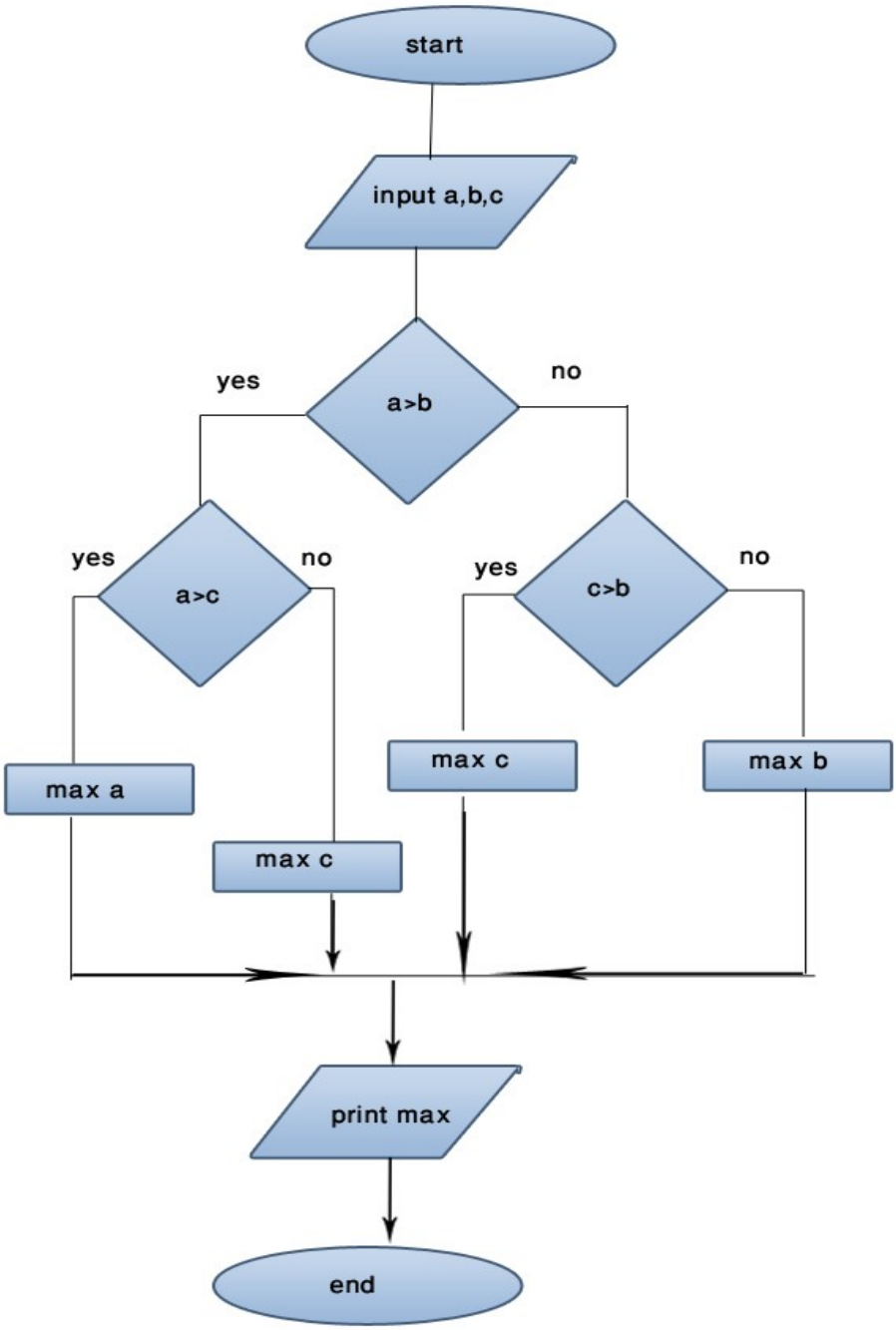
لوفرضنا أن الـ sum=1مثلا فإن هذا سيؤدي إلى خطأ حتماً و السبب في ذلك هو أن الخوارزمية عندما ستبدأ بالحالة 0 سوف ندخل العدد و بالتالي ستجري الخوارزمية عملية جمع مع sum القديمة وهي لو فرضناها 1 لكان الناتج النهائي سيزداد 1 عن القيمة الصحيحة. لذلك فرضنا قيمة الصفر كقيمة ابتدائية مع العلم أن sum=0 هي قيمة الـ sum فقط للحالة 0 و بعدها ستصبح قيمتها من قيمة a المدخل (sum=sum+a=0+a=a)و في الحالة التي تليها تصبح قيمة الـ sum هي

sum=sum+a=a1+a2

التمرين الثالث:

ارسم مخططاً تدفقياً لمعرفة أية الأعداد أكبر من بين 3 أعداد مدخلة

قبل البدء بالحل يجب أن نفكر بالحل الرياضي لها ، سنبدأ في إحدى الاحتمالات باعتبار أن الأعداد هي a، b، c و الآن لنفرض هل a>b إذا كان لا هل c>b إذا كان نعم كانت c هي أكبر الأعداد وإذا كان لا كان b أكبر الأعداد وإذا كان a>b محققة هل a>c إذا كان نعم كانت a هي أكبر عدد و أما إن كان لا كانت c أكبر الأعداد و الآن نرسم المخطط بعد دراسة الحالات .


start
input a,b,c
a>b
yes
no
a>c
yes
no
c>b
yes
no
max a
max c
max c
max b
print max
end

التمرين الرابع :

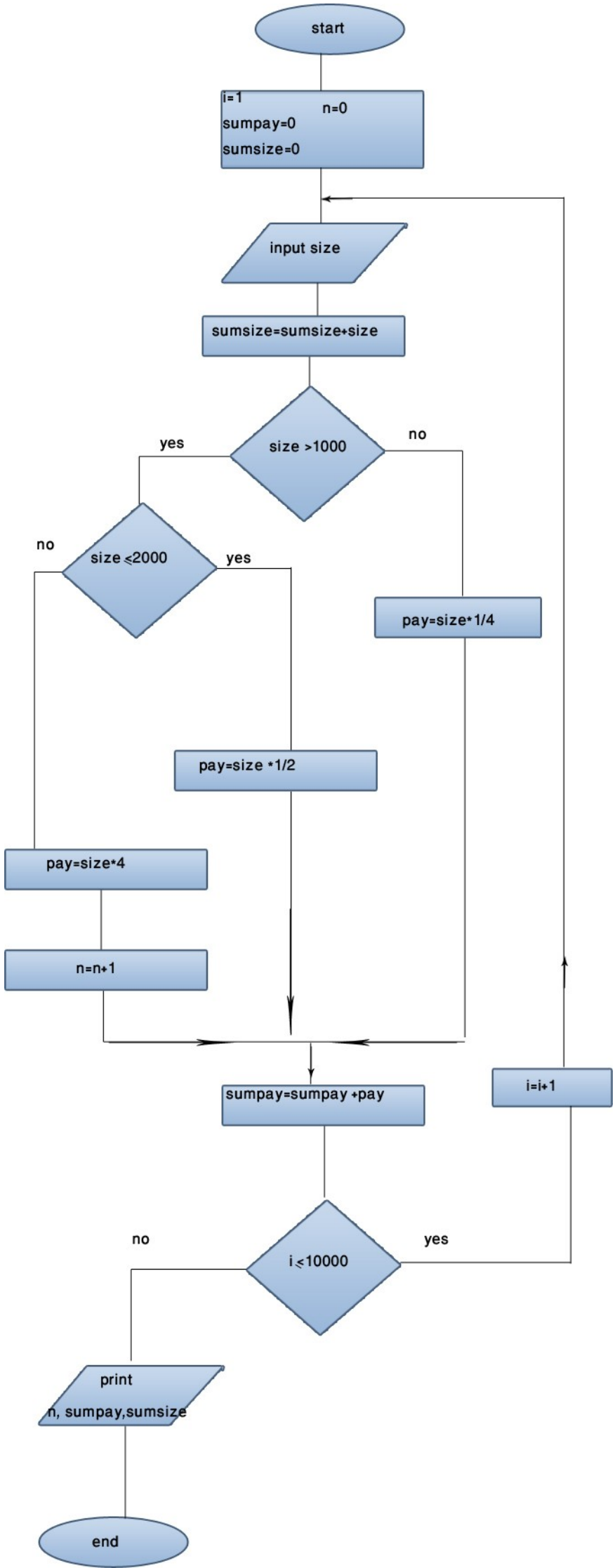
لدى مؤسسة كهرباء 10000 مشترك و ترغب المؤسسة في جباية قيمة فواتير الكهرباء لدورة ما ، فإذا كانت كمية الكهرباء (ولتكن size) أقل أو تساوي 1000 كيلو فإن سعر الكيلو يكون (ربع دينار) و إذا كانت كمية الاستهلاك أكثر من 1000 و أقل أو تساوي 2000 كيلو فإن سعر الكيلو يكون (نصف دينار) عدا عن ذلك فإن سعر الكيلو يكون(أربع دنانير) و المطلوب ضع المخطط التدفقي الذي يتضمن ما يلي :

-قراءة كميات الاستهلاك لكل المشتركين .

-حساب و طباعة قيمة الفاتورة لكل مشترك (لتكن pay) .

- حساب عدد المشتركين الذين يزيد استهلاكهم للكهرباء عن 2000 كيلو في الدورة (و ليكن n).
- حساب المبلغ الإجمالي (و ليكن amount) الذي سيتم تحصيله من قبل المؤسسة خلال الدورة الآنفة الذكر .
- طباعة عدد المشتركين n و المبلغ الإجمالي amount .

الحل:


start
i=1
n=0
sumpay=0
sumsize=0
input size
sumsize=sumsize+size
size >1000
yes
no
size ⩽2000
no
yes
pay=size*1/4
pay=size *1/2
pay=size*4
n=n+1
sumpay=sumpay +pay
i=i+1
i ⩽10000
no
yes
print
n, sumpay,sumsize
end

أولا و بما أن هذه المسالة معقدة أو متشابكة إلى حدّ ما فمن المفضل في هذه الحالات تفريغ المعطيات بشكل مختصر لكي يسهل الحل.
أولا : الرموز
عدد المشتركين n ، و حجم الاستهلاك لكل مواطن size ولكل المواطنين sumsize، الفاتورة pay ، المبلغ الإجمالي sumpay
ثانياً : التعريفات
اكيلو ¼ دينار لمن أقل أو يساوي 1000 و ½ دينار لمن أكبر من 1000 و أقل أو يساوي 2000
1كيلو 4 دنانير لمن أكبر من 2000

تمارين غير محلولة

التمرين الأول: ارسم مخططاً لحساب العاملي لأي عدد $n!$.

التمرين الثاني : ضع المخطط التدفقي اللازم لحساب و طباعة جذور المعادلة

$$ax^2+bx+c$$

و إلى هنا ننهي مقدمتنا حول الخوارزميات و لكن يجب أن ننوه أن بحث الخوارزميات بحث واسع جدا يشمل جميع جوانب الحياة و فيه تشعبات كثيرة و لكن قد استخدمنا منه هنا ما يفيدنا في التمهيد و الدخول إلى البرمجة بلغة الـ سي بلس بلس [1].

سي بلس بلس

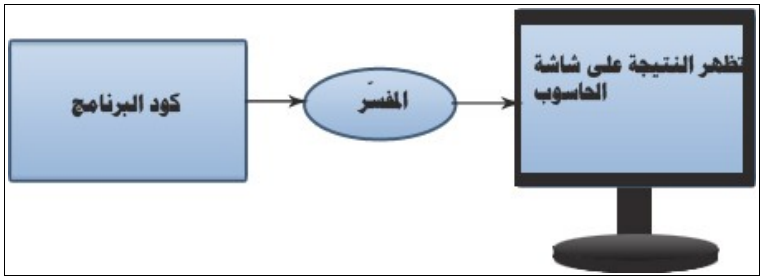
من الجيد التحدث عن تاريخ هذه اللغة و كيف نشأت و تطورت حتى وصلت إلى الشكل الحالي و لكن هذا الكتب ليس كتاب تخصصي و إنما مدخل إلى هذه اللغة الواسعة جداً لذا دعونا نبدأ بتعلّمها بدأ من الأول .

1 المصدر الأساسي لهذه المقدمة مجموعة من المحاضرات د.أكرم مذكور

تعد لغة الـ سي بلس بلس لغة عالية المستوى و هناك العديد من لغات البرمجة الأخرى عالية المستوى مثل الجافا و فورتران و تعرف اللغة العالية المستوى بأنها اللغة ذات التعليمات القريبة من اللغة المحكية و من هذه التسمية لابد ان يخطر إلى بالك أنه هناك لغة منخفضة المستوى و التي تسمى أحياناً لغة الآلة، و يجب على أي برنامج مكتوبة بأي لغة عالية المستوى أن يترجم إلى اللغة منخفضة المستوى و هذا يعد كسلبية صغيرة للغات عالية المستوى لأن هذه الترجمة تحتاج لبعض الوقت حتى تتم .

و هناك طريقتين لقراءة البرامج المكتوبة بلغة عالية المستوى :

1- المفسّر :و هو برنامج يقرأ و يترجم البرنامج سطراً سطراً.[2]

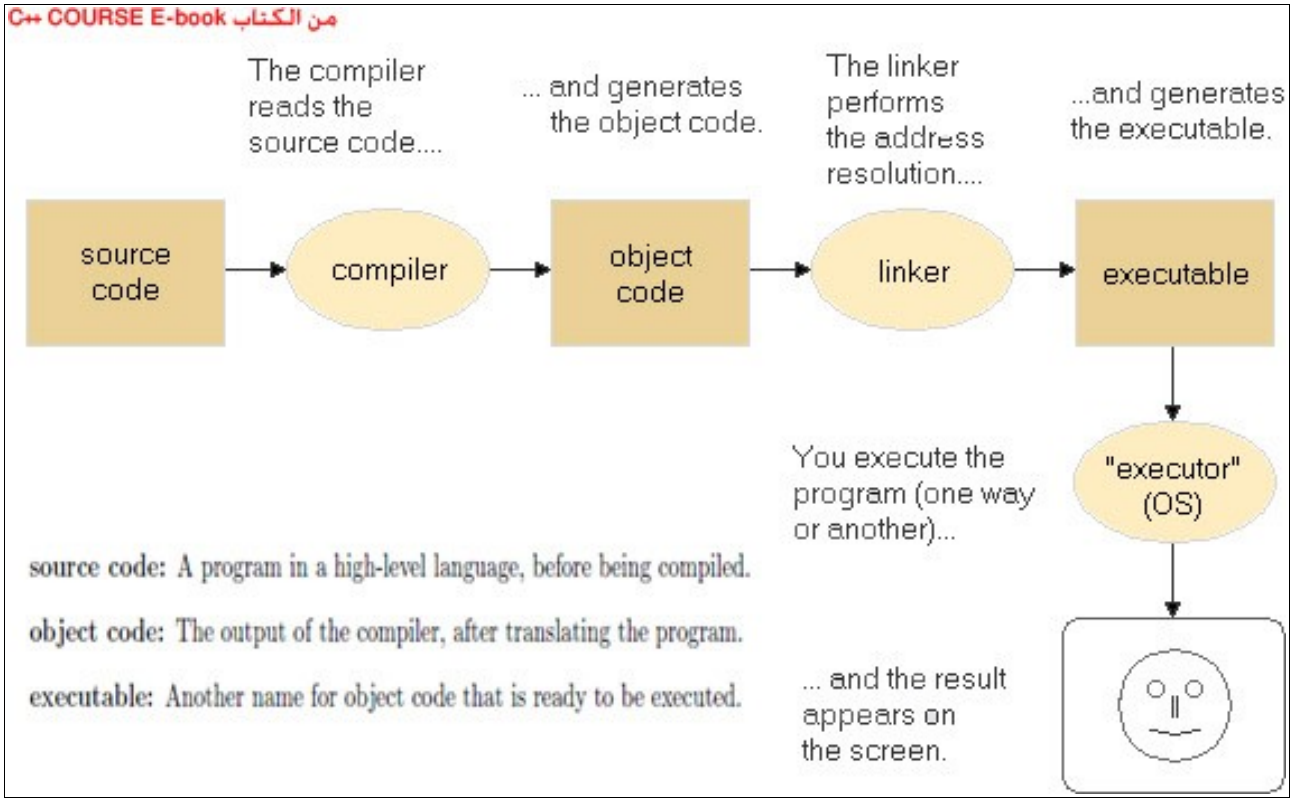


2- المترجم compiler:

يقرأ كود البرنامج بشكل كامل و من أشهر المترجمات البرنامج الذي تنتجه الشركة الاحتكارية مايكروسوفت و المسمى فيجوال استيديو .

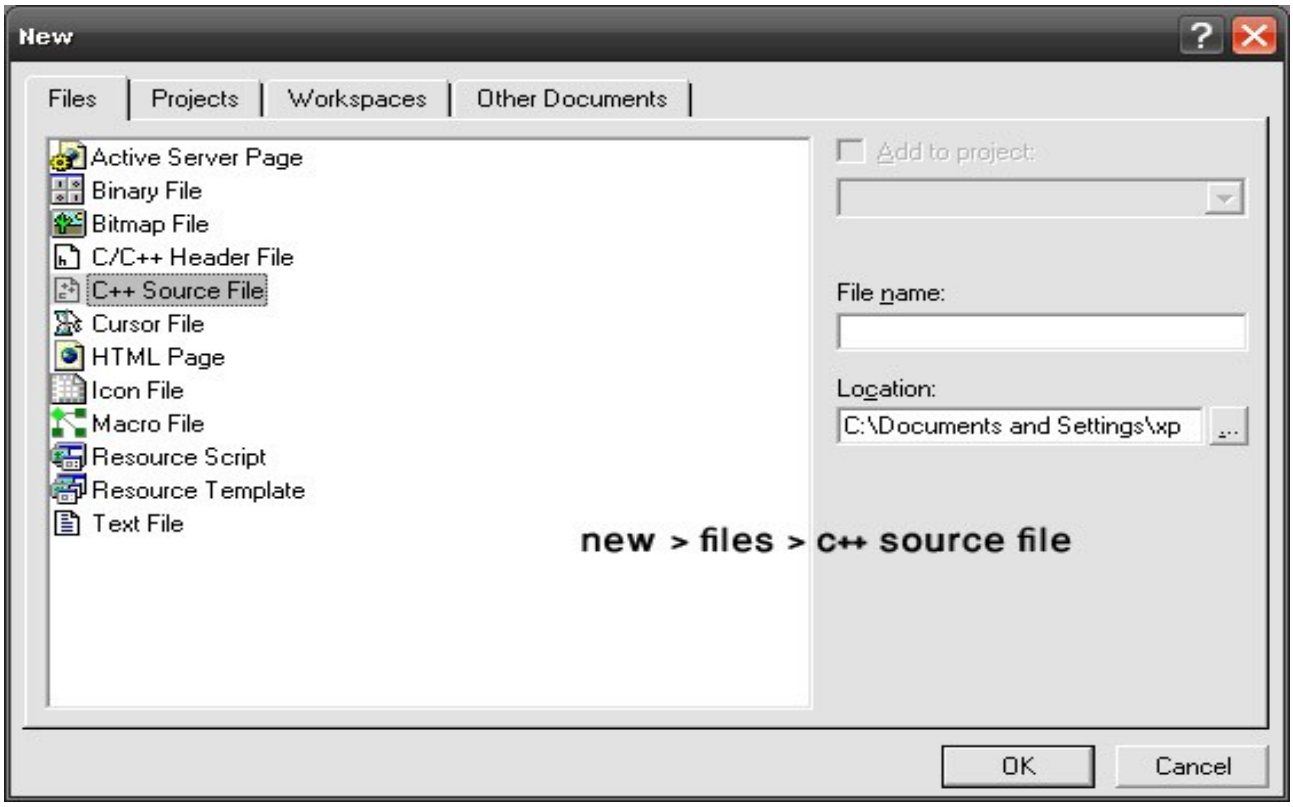
2 C++ cource,e-book

مراحل الترجمة .



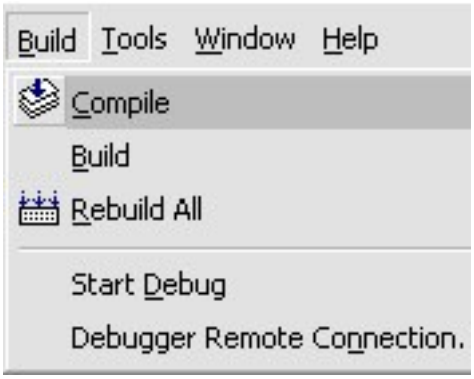
أين نكتب كود البرنامج ؟

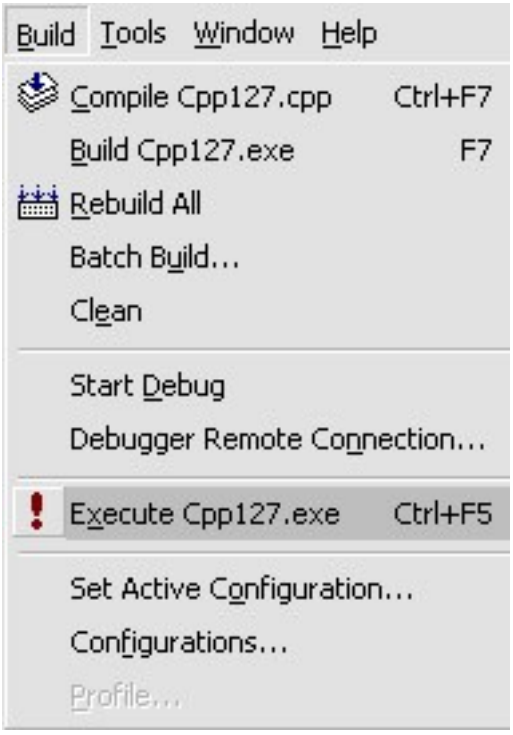
نكتب الكود على أي مترجم و من ثم نقوم بالأمر compile و ننفذ البرنامج، تابع تسلسل الخطوات في الصورة التالية للبرنامج آنف الذكر


New
Files
Projects
Workspaces
Other Documents
Active Server Page
Binary File
Bitmap File
C/C++ Header File
C++ Source File
Cursor File
HTML Page
Icon File
Macro File
Resource Script
Resource Template
Text File
Add to project:
File name:
Location:
C:\Documents and Settings\xp
new > files > c++ source file
OK
Cancel

```
//generate some simple output
#include<iostream.h>
void main()
{
cout<<"this is my first programme"<<endl ;
}
```

نكتب البرنامج





البرنامج الأول :

اكتب برنامجاً بلغة الـ سي بلس بلس يظهر للمستخدم العبارة التالية

this is my first programme

الخرج	

this is my first programme
Press any key to continue
	//generate some simple output
	#include<iostream.h>
	void main()
	{
	cout<<"this is my first programme"<<endl ;
	}

شرح هذا البرنامج :

هناك بعض الأمور التي يأتي شرحها في مراحل متقدمة نوعاً ما مثل السطر الثاني و لكن نكتفي الآن كبداية بمعرفة الأمور التالية :

1-إن الشكل العام للبرنامج في هذه اللغة

#include<iostream.h>(ملف التوجيه الخاص بالإدخال الإخراج)

```
void main() (هناك أنواعاً أخرى و لكن نستخدم هذا النوع كبداية)
{
statments;(التعليمات)
}
```

ملفات التوجيه(head files):

الملفات المصدرية ذات اللاحقة cpp ليس هي الوحيدة الملحوظة في البرامج و لكن هناك نوع آخر من الملفات و هي ملفات التوجيه head files و هي لديها لاحقة h و الغرض منها هي تضمين الملفات التي نريد استعمالها .[3]

و لولا تضمين مكتبة الدخل و الخرج لما كان لاستخدام cout -و هي قرار الإخراج -أي فائدة لأنه لن يتم التعرّف عليها إلا من خلال المكتبة iostream.h التي تحوي تعريفاً لها

2- إيّ ملاحظة أو شرح تريد إضافته على كود البرنامج يمكن إضافته بالشكل التالي :

```
//adde any comment you want here
```

أو

```
/*adde any comment you want here*/
```

3-إن التعليمة endl تنهي لك السطر و تبدأ بسطر جديد في الإظهار و مثال على ذلك البرنامج التالي اكتب برنامجاً يظهر لك الرسالة التالية في حالتين :

```
-my name is ahmad .i'm a student
-my name is ahmad.
I'm a student
```

الحل:

```
//a programme shows the job of endl
statment
#include<iostream.h>
void main()
{
cout <<"my name is ahmad .";
cout <<"i'm a student ";
}
```

الخرج

```
my name is ahmad .i'm a student Press any key
```

الحالة الثانية :

```
//a programme shows the job of endl
statment
#include<iostream.h>
void main()
{
cout <<"my name is ahmad ."<<endl;
cout <<"i'm a student ";
}
```

الخرج

```
my name is ahmad .
i'm a student Press any key
```

4- إن التعليمة cout تظهر لك على الشاشة كل ما يوجد بين علامتي التنصيص" " . و الجدير بالملاحظة أن التعليمة تكتب بالشكل التالي و من الخطأ أن تقلب اتجاه الأسهم و السبب سيمرّ معنا بعد قليل .

```
Cout<<" test here"
```

قواعد بناء البرنامج في سي بلس بلس

-يبدأ البرنامج بالعبارة

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

-يتكون البرنامج من دالة رئيسية main تبدأ بـ{ وتنتهي بـ} .
-كل عبارة برمجية يجب أن تنتهي بفاصلة منقوطة مع وجود بعض الاستثناءات سوف نتعرّف عليها

تباعاً .

ملاحظة :

لا يشترط أن تكتب البرنامج في هذه اللغة على عدة أسطر بل يمكنك أن تكتب البرنامج كله على سطر واحد شرط أن تضع الفواصل و الأقواس اللازمة و لكن من العادات البرمجية الجيدة أن يكون برنامجك واضحاً للقارئ مزود بمجموعة من الملاحظات موزعاً على أسطر .

التمرين الثاني :

اكتب برنامجاً بلغة السي بلس بلس يطلب من المستخدم أن يدخل اسمه و عمره ثم يظهر له اسمه و بجانبه عمره .

```
//declaring variables
#include <iostream.h>
void main ()
{
char name;
int age;
cout <<"enter the first letter of your
name .please!"<<endl ;
cin>>name ;
cout <<"how old are you ?"<<endl ;
cin>>age;
cout <<"the first letter of your name
"<<name<<"---your age is "<<age<<endl ;
}
```

الخرج

```
my name is ahmad .
i'm a student Press any key
```

الحل:

الشرح :

إن المفاهيم الجديدة التي أوردناها في هذا البرنامج هي : الإدخال و استخدام التصريح عن المتغيرات و إخراج قيم المتغيرات .

التصريح عن المتغيرات :

عندما تصرّح عن المتغير بقولك على سبيل المثال

int a ;

تكون قد حجزت مكان في الذاكرة مخصّص لهذا المتغير

ملاحظة : الحالة العامة تقول أن قيمة المتحول متوافقة مع نوعه مع الإمكانية في بعض الأحيان التحويل من نوع إلى آخر و سيمرّ لاحقاً مثال على هذا إن شاء الله تعالى .

أنواع المتغيرات :

إن المتغيرات التي ستمر معنا في هذا الكتاب هي الصحيحة int و الحقيقية float و الحرفية char و الباقي يوضحها الجدول التالي :[4]

Name	Description	Size*	Range*
char	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean value. It can take one of two values: true or false.	1byte	true or false
float	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	Wide character.	2 *or* 4 bytes	1 wide character

4 هذا الجدول من كتاب C++ Language Tutorial cplusplus.com

المتغير البولي:

Boolean variable يأخذ هذا المتغير قيمتين 0 و هي دلالة للخطأ و 1 دلالة لصحيح

مثلا في هذا البرنامج البسيط :

```
#include<iostream.h>
void main()
{
bool test1,test2;
test1=true;
test2=false;
cout<<"test1= "<<test1<<"
test2="<<test2<<endl;

}
```

الخرج

```
test1=1
test2=0
press any key to continue.
```

و تستخدم المتغيرات البولية كثيراً في العبارات الشرطية و ليس من المهم الآن أن تستوعب البرنامج التالي- الذي يتحقق من تساوي عددين مدخلين - لأنه يحتوي على بعض التقنيات التي لم نأخذها بعد و لكن الأهم هي الفكرة

```
#include<iostream.h>
void main()
{
int x,y;
cout<<"enter 2 numbers "<<endl;
cin>>x>>y;
bool isequal=(x==y);
cout<<isequal<<endl;
isequal=equal(x,y);
if(isequal)
cout<<"equal"<<endl;
else
cout<<"not  equal"<<endl;
}
```

الخرج

```
enter 2 numbers
6
3
0
not  equal
Press any key to continue
----------------------------------------
enter 2 numbers
3
3
1
equal
Press any key to continue
```

حيث أننا أسندنا للمتغير البولي عبارة علائقية و لها حالتين محققة 1 أو غير محققة 0 و عندما استخدمنا المتغير في الشرط if فإنه عند تحققه فإن البرنامج يطبع equal و في حال غير ذلك يطبع not equal

الإدخال:

نستخدم التعليمة cin لإدخال القيم (حرفية كانت أو عددية) و هي من الشكل التالي حيث a متغير مصرّح عنه :

```
cin>>a ;
```

و هنا يجب الإشارة أن اتجاه الأسهم هذا لو عُكس أصبح اتجاه أسهم التعليمة cout و بالتالي أصبح خطأ برمجي .

إخراج قيم المتغيرات باستخدام التعليمة cout:
كما ورد معنا في الكود السابق

```
cout <<"the first letter of your name
"<<name<<"---your age is "<<age<<endl ;
```

استخدامنا التعليمة cout لإظهار العبارات النصية المحصورة بين "" أو قيم المتغيرات و التي نضعها دون إشارتي التنصيص .

البرنامج الثالث :
اكتب برنامجاً بلغة السي بلس بلس تظهر فيه الوقت الحالي على حاسوبك الشخصي ممثلاً صباحا بـ a أو مساءً بـ p .
الحل :

```
//using Assignment
#include<iostream.h>
void main()
{
int min,hour,second ;
char time ;
hour=4;min=51;second=52;time=' p';
cout <<"now it is
"<<hour<<":"<<min<<":"<<second<<"."<<time
<<endl;
}
```

الخرج

```
now it is 4:51:52.p
Press any key to continue
```

الاسناد:

عندما نريد أن نسند قيمة ما إلى متغير يجب أن أن نصرّح عن ذلك بصيغة تختلف بين الإسناد الحرفي و الرقمي
الاسناد الرقمي:
وفق الصيغة

```
hour=4;min=51;second=52
```

الاسناد الحرفي:

```
time='p'
```

البرنامج الرابع :
اكتب برنامجاً بلغة السي بلس بلس يطلب من المستخدم إدخال عدد الساعات التي نامها البارحة و يحسب له ما يلي :

- كم دقيقة نام .
- ما هي عدد الساعات الزائدة/الناقصة عن الحد اللازم .

الحل :

```
//using operation in c++
#include<iostream.h>
void main()
{
cout <<"how many hours did you sleep
lastnight? "<<endl;
int hours,min ,m_P;
cin>>hours;
min =hours*60;
m_p=hours-8 ;
cout <<"you slept last night"
<<min<<"minute"<<endl;
cout <<"and you are over the normal ratio by
"<<m_p<<"hours"<<endl ;
}
```

الخرج

```
?how many hours did you sleep lastnight
                6
    you slept last night360minute
 and you are over the normal ratio by
              -2hours
      Press any key to continue
```

الشرح :

هذا البرنامج يسمح لك بالتعرّف على العمليات الأساسية في هذه اللغة .

جدول يبّن أهم رموز العمليات الحسابية و المنطقيةو المنطقية في اللغة:

العملية	الرمز الرياضي	الرمز البرمجي	مثال
الجمع	+	+	a+b

الضرب		×	*	a*b
القسمة		÷	/	a/b
الطرح		−	−	a-b
باقي القسمة الصحيح		mod	%	a % b
المساواة		=	==	a==b
عمليات علائقية Relational operators	عدم المساواة	≠	!=	a!=b
	أكبر أو تساوي	≥	>=	B >= A
	أصغر	<	>	$a<b$
	أكبر	>	>	$a>b$
	أصغر أو تساوي	≤	<=	A <= b
عمليات منطقية logical operation	نفي		!	!(x==1)
	و	∧	&&	(y!=0)&&(x==1)
	أو	∨	||	(y==6)||(x>3)

مثال[5]

5 c++ language tutorial

```
(7 == 5)     // evaluates to false.
(5 > 4)      // evaluates to true.
(3 != 2)     // evaluates to true.
(6 >= 6)     // evaluates to true.
(5 < 5)      // evaluates to false.
```

```
(a == 5)     // evaluates to false since a is not equal to 5.
(a*b >= c)   // evaluates to true since (2*3 >= 6) is true.
(b+4 > a*c)  // evaluates to false since (3+4 > 2*6) is false.
((b=2) == a) // evaluates to true.
```

البرنامج الخامس :
افترض أنه طلب منك برمجة برنامج يطلب من الزائر لأحد الألعاب في مدينة الملاهي إدخال طوله لتحديد إمكانية دخوله لهذه اللعبة أم لا علماً أنه إذا كان طوله أقل من 160 سم لا يسمح له و باقي الأطوال يسمح لها .

```
#include<iostream.h>
void main ()
{
cout <<"how tall are you ?"<<endl ;
int tall ;
cin>>tall ;
if(tall<=160)
cout <<"sorry you can't enter. you are under
160 c.m"<<endl ;
else
cout<<"welcom !!"<<endl ;
}
```

الخرج

```
how tall are you ?
162
welcom !!
Press any key to continue
```

الشرح

1-بنية التحكم if :

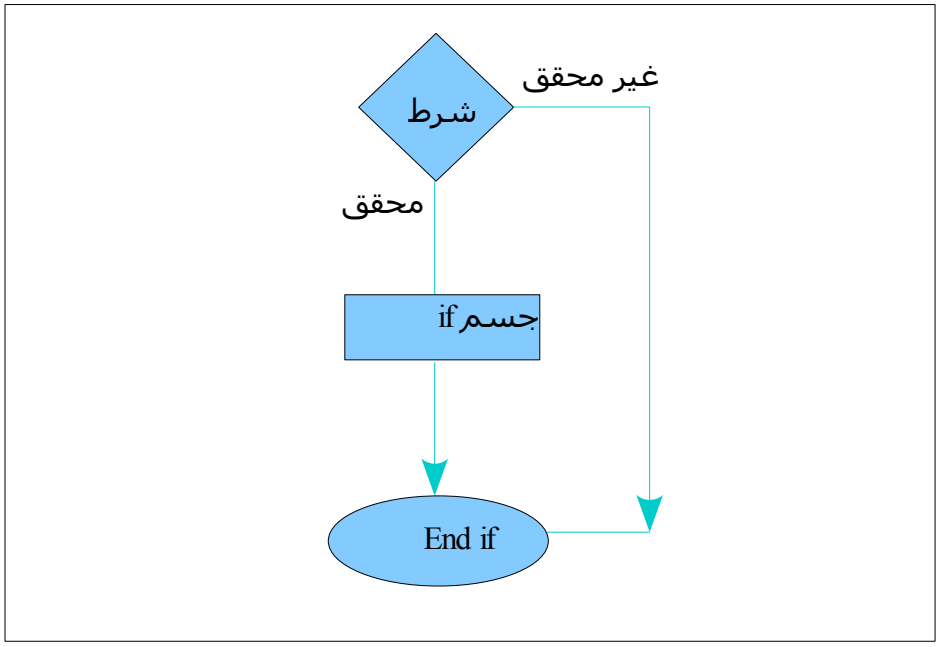
الشكل العام لبنية التحكم if

```
if(condition)
statement
```

أو

```
if(condition)
{
block of statements
}
```

عمل if باستخدام المخططات التدفقية



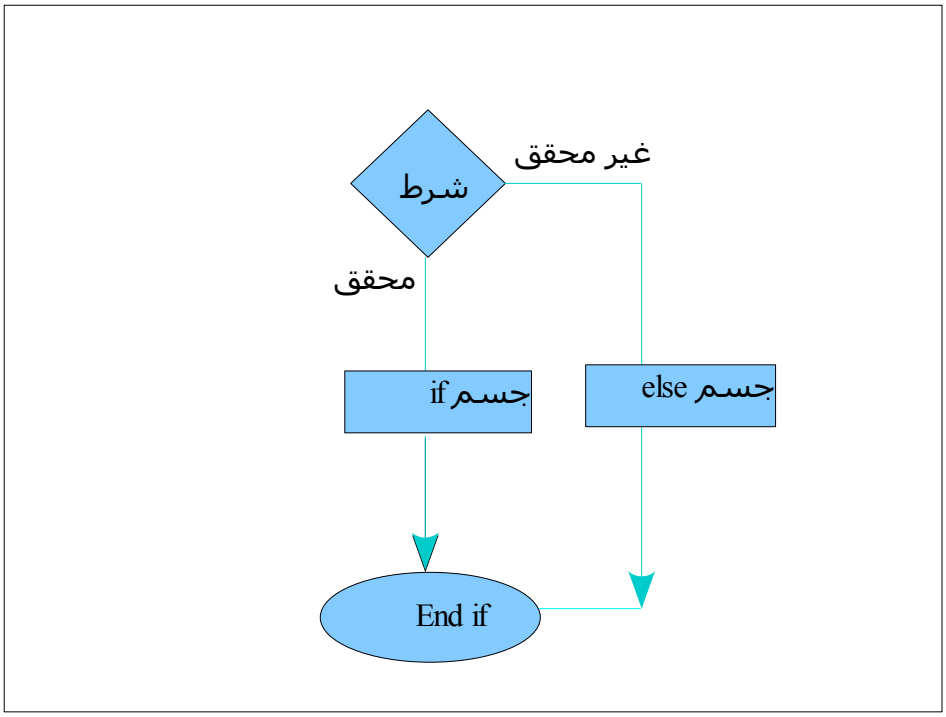
2- بنية التحكم الثانية if-else:

الشكل العام لبنية التحكم هذه

```
if (condition )
statement
else
statement
```

```
if(condition)
{
block of statements
else
{
block of statements
}
```

عمل eles-if باستخدام المخططات التدفقية



3-بنية التحكم الثالثة if else if :

الشكل العام لهذه البنية

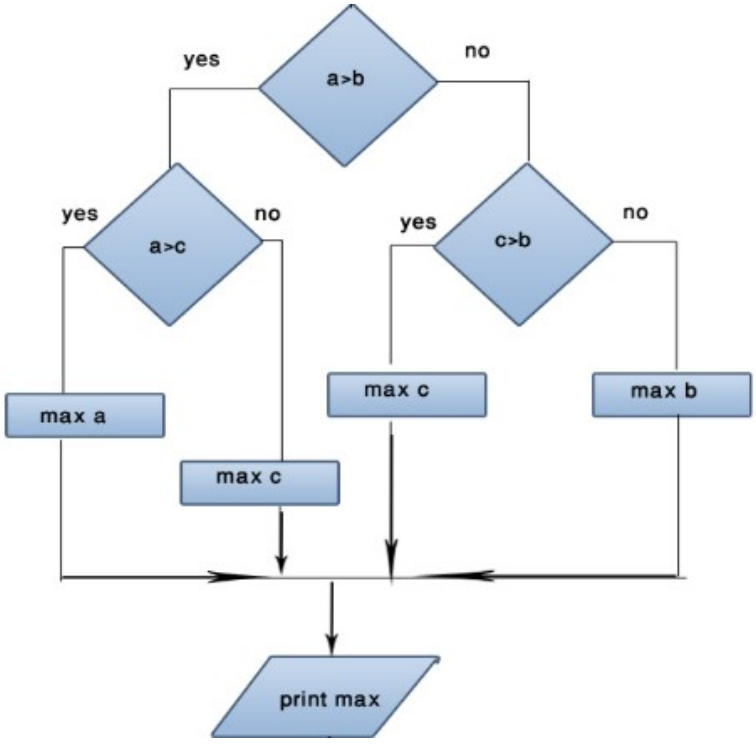
```
if(condition1)
statement1
if else(condition2)
statement2
else
statement3
```

فلو تحقق الشرط الأول نُفذ statement1 ولو تحقق الشرط الثاني نُفّذ statement2 و إن لم يتحقق الشرطان يتم تنفيذ statement3

```
if(condition1)
{
block of statements1
}
else if(condition2)
{
block of statements2
}
else
{
block of statements3
}
```

البرنامج السادس :
اكتب برنامجاً بلغة السي بلس بلس يحدد العدد الأكبر بين ثلاث أعداد مدخلة :

الحل :
ربما تذكرون في مقدمة الكتاب استخدما المخططات التدفقية لحل مثل هذه الحالات و الآن دعونا نشاهد الجزء من المخطط التدفقي الذي وضعناه لحل نفس المسألة لتجدوا السهولة في كتابة الكود المصدري لهذا البرنامج بالنظر إلى الشكل .



```
//using if else if  control structure
#include<iostream.h>
void main()
{
int a,b,c;
cout <<"enter the first number"<<endl ;
cin>>a;
cout <<"enter the second number"<<endl;
cin>>b;
cout <<"enter the third number "<<endl;
cin>>c;
if(a>b)
{//start of first if
if (a>c)
cout <<"the biggest number is"<<a<<endl;
else
cout <<"the biggest number is"<<c<<endl;
}//end of first if
else if(c>b)
{//start of else if
if (c>b)
cout <<"the biggest number is"<<c<<endl;
else
cout <<"the biggest number is"<<b<<endl;
}//end of else if
}//end of the programme
```

الخرج

```
enter the first number
2
enter the second number
5
enter the third number
9
the biggest number is9
Press any key to continue
```

البرنامج من 1 إلى 11 قابلة للحل حتى الآن من الجزء العملي

البرنامج السابع :
اكتب كود في لغة السي بلس بلس يعمل كآلة حاسبة تقوم بالعمليات الحسابية الأساسية (الجمع، الطرح ،الضرب، القسمة)
الحل:

```
//basic calculator in c++
#include<iostream.h>
void main()
{
char operation;
cout <<"enter the sympole of the
operation"<<endl ;
cin>>operation;
int a,b,r ;
cout <<"enter the two numbers "<<endl ;
cin>>a>>b;
switch(operation)
{
case'+':r=a+b;
break;
case'*':r=a*b;
break;
case'/':r=a/b;
break;
case'-':r=a-b;
break;
}
cout<<"result= "<<r<<endl ;
}
```

الخرج

```
enter the sympole of the operation
+
enter the two numbers
3
6
result= 9
Press any key to continue
```

ونلاحظ أننا وضعنا بعد كل case التعليمة break ربما سيجيبك الشكل التالي و الذي يمثّل الخرج التفيذي عندما حذفنا break في الحالة الأولى

```
enter the sympole of the operation
+
enter the two numbers
9
5
result= 45
Press any key to continue
```

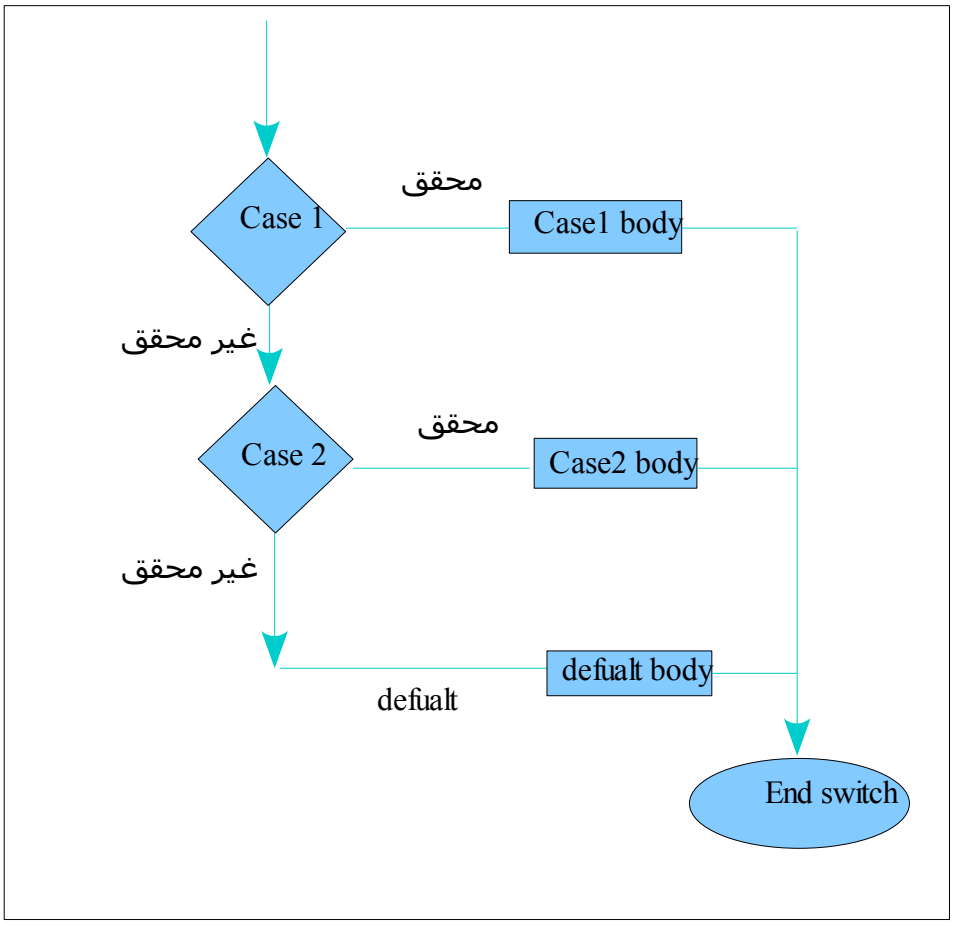
وهذا يدلنا أن البرنامج قام أولا بالعملية التالية r=a+b=9+5 ولكن لم يجد التعليمة break فقام بالحالة الثانية و هي:r=a*b=5*9و هذا يدلنا أن التعليمة break تفيد في الخروج من بنية التجكم switch.

4-بنية التحكم switch:

الشكل العام

```
switch(expretion)
{
case value 1 : statement1;
break ;
case value 2 : statemen2;
 break  ;
case value n : statementn;
 break;
}
```

عمل الـ switch باستخدام المخططات التدفقية



ملاحظة :
يمكن أن نضع التعليمة default في نهاية البنية switch و التي تحوي على الحالة التي يقوم بها البرنامج في عدم تحقق أي حالة من الخالات السابقة كما يلي :

```
switch(operation)
{
case'+':r=a+b;
break ;
case'*':r=a*b;
break;
case'/':r=a/b;
break;
case'-':r=a-b;
break;
default : cout <<”this operation is not supported”<<endl ;
}
```

البرنامج الثامن :
اكتب برنامجاً يظهر لك الأعداد من واحد إلى عشرة .

```
//using looping structure(for)
#include<iostream.h>
void main ()
{
int i;
 for(i=1;i<11;i++)
{
cout<<i<<endl ;
}
}
```

الخرج

```
1
2
3
4
5
6
7
8
9
10
Press any key to continue
```

الشرح :

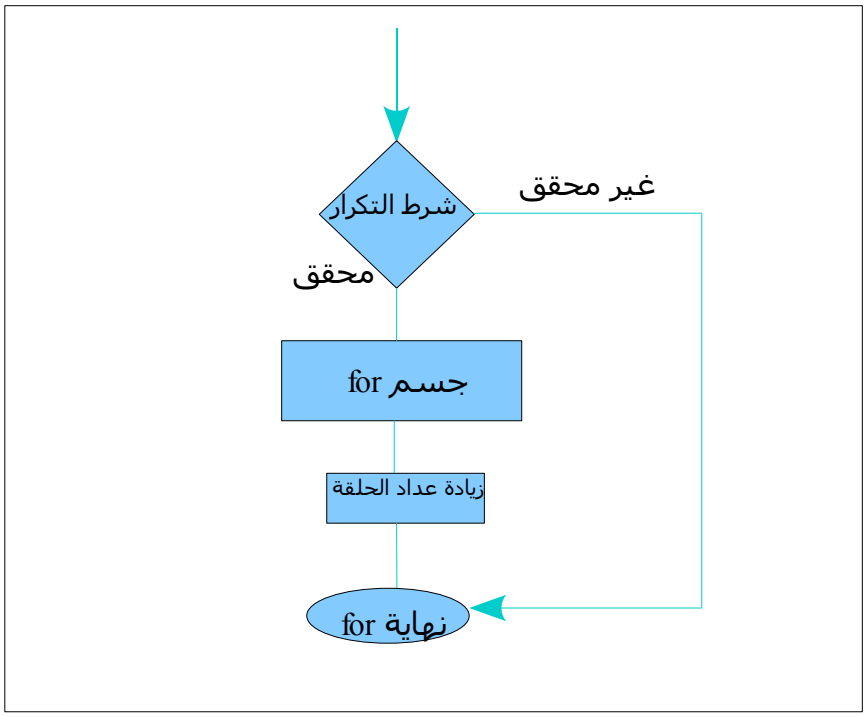
بنى التكرار:

1-بنية التكرار for

الشكل العام :

```
for(first value;condition ; increas or decreas)
{
statement/s
}
```

عمل for باستخدام المخططات التدفقية:



2-بنية التكرار while

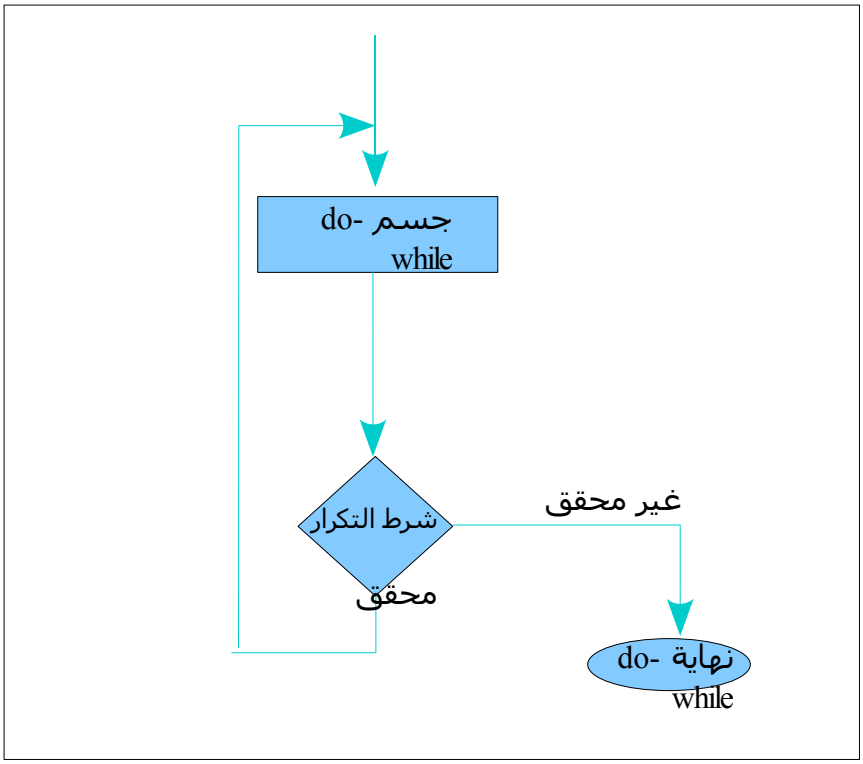
الشكل العام:

```
while(condition)
{
statement/s;
increas or decreas
}
```

3-بنية التكرار do while

```
do
{
statement/s
increas or decreas
}
while(condition);
```

عمل do-while من خلال المخططات التدفقية

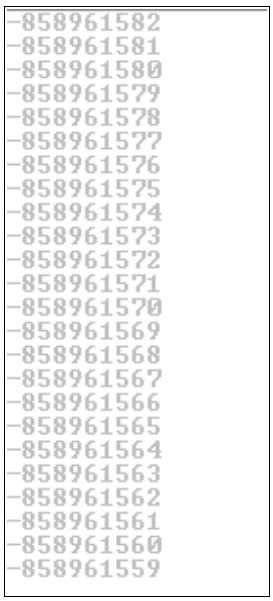


مثال:

```
//using looping structure(do while)
#include<iostream.h>
void main ()
{
int i=1;
 do
{
cout<<i<<endl ;
i++;
}
while(i<11);
}
```

ملاحظه: لقد حددنا قيمة ابتدائية للمتغير i في المثالين السابقين و لذلك لأنه في حال عدم تحديد قيمة للمتغير ابتدائية فإن الحاسوب سوف يأخذ قيمة عشوائية .

مثال :
احذف من الكود السابق القيمة الابتدائية للمتحول i و لاحظ أن الحاسوب سوف يبدأ بإظهار أرقام سالبة صغيرة جداً و يبدأ بالإضافة إليها واحد حتى يصل إلى 11 لينتهي البرنامج



البرنامج التاسع:

اكتب برنامجاً يكتب الأحرف من a إلى الحرف g مع عدم إظهار الحرف b من ضمنهم

```
//continue statement
#include<iostream.h>
void main()
{
for(char a='a';a<='g';a++)
{
      if(a=='b')
            continue;
cout<<a<<endl ;
}
}
```

الخرج

```
a
c
d
e
f
g
Press any key to continue
```

ملاحظة:

if(a=='b')

هذا شرط و ليس إسناد قيمة .

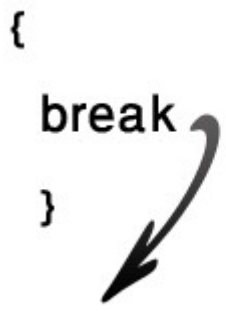
البرنامج العاشر:
اكتب برنامجاً بلغة السي بلس بلس يقوم بطباعية الأعداد من 1إلى 5 حيث يتوقف باستخدام break:

```
//using break statement
#include<iostream.h>
void main()
{
int a=1;
for(a=a;a<6;a++)
{
if(a==6)
break;
cout<<a<<endl ;
}
}
```

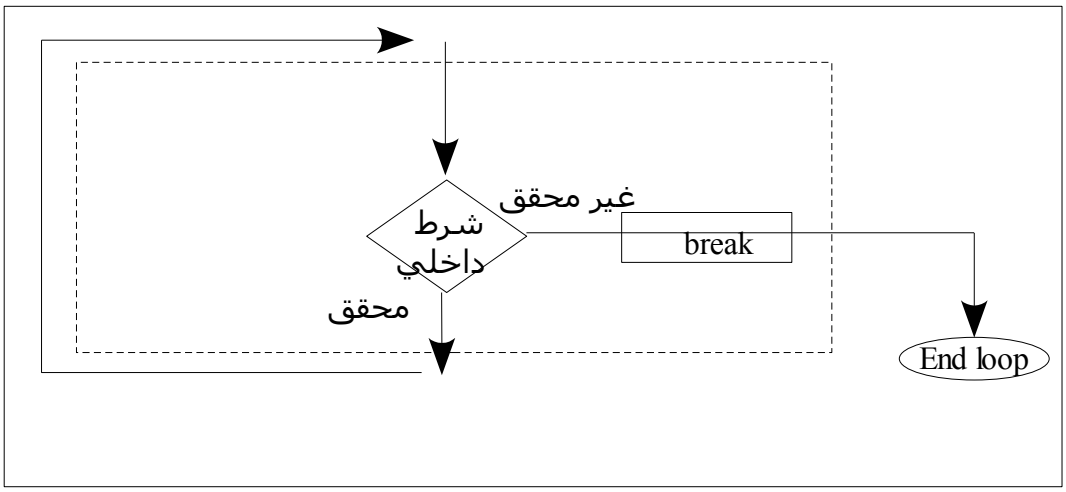
```
الخرج
  1
  2
  3
  4
  5
Press any key to continue
```

البنيتان break و continue:

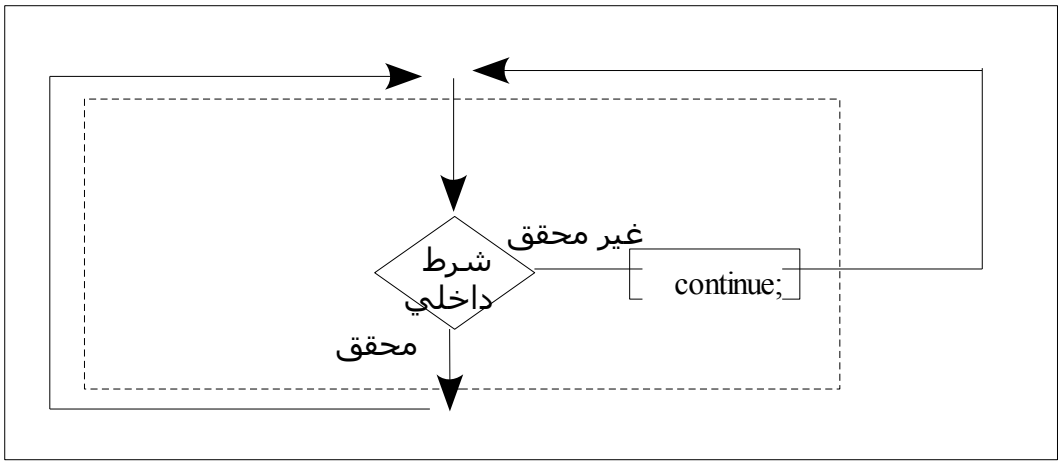
تُستخدم التعليمتان breakو continue من أجل تغيير مجرى تدفق التحكم ضمن البرنامج . حيث تسبب التعليمة break عند تنفيذها مع إحدى البنى التالية switch, for , while ,do while. الخروج من هذه البنية و يتابع البرنامج بعدها تنفيذ أول تعليمة تلي البنية و يوضح ذلك الشكل التالي:



و باستخدام المخططات التدفقية



أما التعليمة continue تسبب عند تنفيذها مع إحدى البنى السابقة (ماعدا switch لأن القرار continue لا يستخدم مع البنية switch)
تجاوز تنفيذ التعليمات داخل الحلقة و ذلك في حالة محددة و يتابع حلقة التكرار من أجل المرة التالية ،
و باستخدام المخططات التدفقية



المصفوفات :

و هي وسيلة تخزين أساسية حيث تخزن مجموعة من المتحولات من نوع البيانات واحد ضمن كتلة واحدة داخل الذاكرة و تفيد استخدام تقنية المصفوفات في عدة أمور:

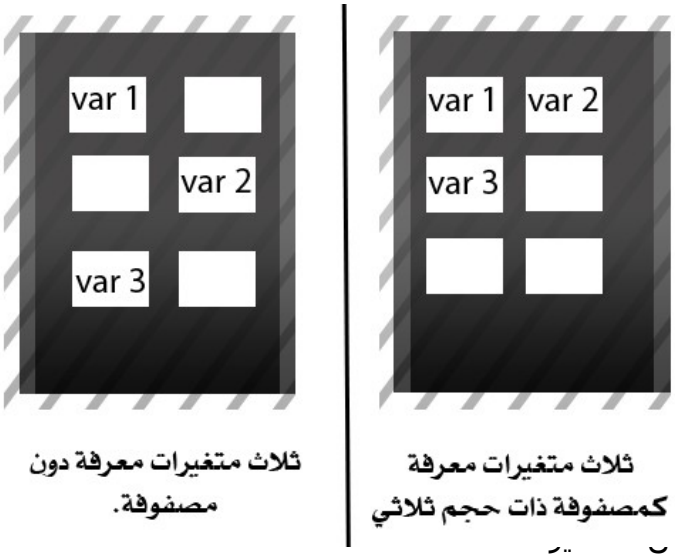
1-اختصار وقت التنفيذ :

لنفرض أن البرنامج يستخدم ثلاث متغيرات var1.var2,var3 وعرفناها دون استخدام مصفوفة فإنه سيتم حجز مواقع لهم في الذاكرة دون أن تكون متتالية بالضرورة

و بالتالي باستخدام المصفوفات يتم اختصار مدة زمنية في التنفيذ و هذا طبعاً لا يلاحظ إلا في البرامج الكبيرة جداً .

ملاحظة: قد يصدف و يتم حجز أماكن متتالية لمتغيرات تم تعريفها دون مصفوفة و لكن ليس دائماً .

و هذا شكل توضيحي و الذي لا يمثّل أبداً الذاكرة على حقيقتها و لكن لتقريب الفكرة لا أكثر .



2-التعامل مع عدد كبير

لو فرضنا أن برنامجنا المكتوب في هذه اللغة احتاج إلى 100 متغير من نوع ما ، فإنه من غير المعقول أن أعرف 100 متغير و كل واحد من هذه المتغيرات باسم مختلف ، لذلك يكفي أن أعرف مصفوفة لها 100 عنصر .

فهم آلية الترتيب التصاعدي أو التنازلي في المصفوفات

نكتب كود الترتيب و من ثم نشرحه إن شاء الله :

```
int a[4]={1,2,3,4},temp;
             for(int i=0;i<3;i++)
             {
                    for(int j=0;j<3-i;j++)
                    {
                           if(a[j]<a[j+1])
                           {
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
                           }
                    }
             }
```

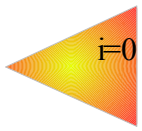
انظر في المصفوفة التالية و التي يراد ترتيبها من الصغير إلى الكبير

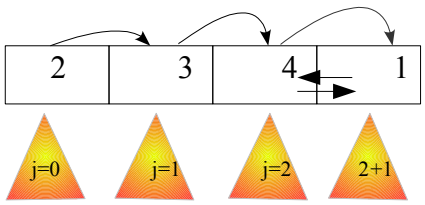
2	3	4	1

و قبل أي شرح لابد لك أن نلاحظ أننا نحتاج حلقتي تكرار متداخلتين لأننا سنقارن كل عنصر مع باقي العناصر و لذلك قد أخذنا i و j و أما بالنسبة لحدود هذين المغيرين فسوف أثبتُ لك بالتجربة لماذا أخذنا الـ i من 0 إلى arrsize-1 و الـ j من 0 إلى arrsize-1-i وبالتالي بالنسبة لهذا المثال لدينا الحدود هي

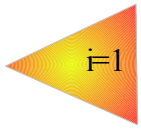
0<i<3 i++

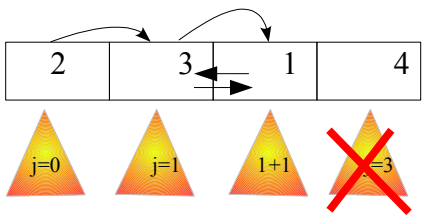
0<j<3-i j++



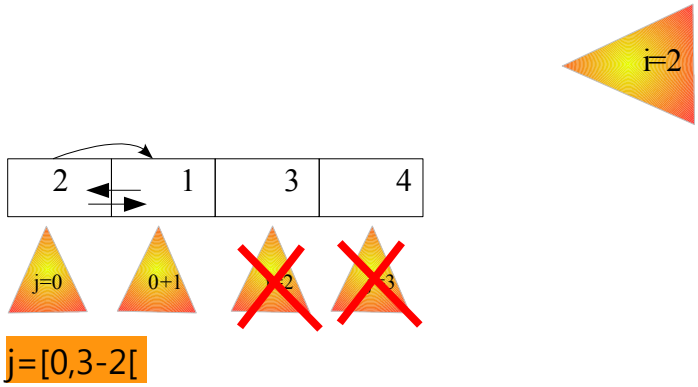


j=[0,3-0[





j=[0,3-1[



و بالتالي أنتهت الحلقة و النتيجة النهائية هي:

1	2	3	4

المؤشرات pointers :

تُرى هل تسائلت لماذا نأخذ أول عنصر من المصفوفة بدليل 0 أي arr[0] و لماذا لم نأخذه 1 ؟
الجواب بالحقيقة مرتبط بالمؤشرات وشرح المؤشرات ليس الصعوبة بقدر ماهو القدرة على توصيل فكرته وبالتالي لنبدأ على بركة الله :

المؤشر :

هو متغير يُخزَّن فيه عنوان جزء الذاكرة الذي يحتله المتغير في الذاكرة.
إذا فإن هذا المؤشر يحمل عنوان المتغير .

هل للمؤشر نوع بيانات ؟

لا يمكن القول أن للمؤشر نوع بيانات كباقي المتغيرات التي يتم تخزين قيمة فيها و لكن هو يشير إلى متغير محرفي أو صحيح أو حقيقي .

التصريح عن المؤشر :

```
pointee_datatype *pointername ;

نوع المتغير الذي يشير إليه *اسم المتغير;
```

تمهيد المؤشر:

يجب أن نقوم بتمهيد المؤشر بعد أو أثناء التصريح عن المؤشر و يجب أن يكون التمهيد بقيمة صحيحة و ليست عشوائية مثل المتغيرات العادية و يستخدم عامل الموضع في التمهيد .

مجالات استخدام المؤشرات :

1- التعامل مع قيمة المتغير مباشرة :
و ذلك باستخدام عامل المواربة * ملحوقاً باسم المؤشر فلو نفذنا الكود التالي كان الناتج :

```
int a=5;
int *p=&a;
cout<<*p<<endl;
```

الخرج:

5

2-التخصيص الديناميكي للذاكرة dynamic allocate:
و يعني تخصيص مساحة من الذاكرة أثناء عمل البرنامج وهذا يمكن اسقاطه على استخدام المصفوفات ، حيث أنه دائماً ما اضطررنا إلى تعيين حجم المصفوفة قبل تنفيذ البرنامج و لكنه الآن و باستخدام المؤشرات فإننا تستطيع القيام بذلك أثناء عمل البرنامج و ذلك باستخدام العامل new و delete :

التركيب النحوي للعامل new:

```
pointee_datatype *pointername ;

pointername=new datatype[size];
```

التركيب النحوي للعامل delete :

```
Delete pointername;
```

و يستخدم العامل delete من أجل تحرير الجزء الذي تم تخصيصه عبر العامل new و إعادته إلى نظام التشغيل .

مثال :

```
#include<iostream.h>
void main()
{
      int nwsize;
      cout<<"enter the size : \n";
      cin>>nwsize;
      int *ptr;
      ptr=new int[nwsize];
      *ptr=5;
      cout<<"test(1)befor delete:"<<*ptr<<endl;
      delete ptr;
      cout<<"test(2)after delete:"<<*ptr<<endl;
                  }
```

```
الخرج
enter the size :
25
test(1)befor delete:5
test(2)after delete:-572662307
```

كيف أتعامل مع ما تم تخصيصه باستخدام العامل new ?

نقوم بتحريك المؤشر باستخدام الزيادة بواحد مثلاً للعنصر الأول و اثنين للثاني و هكذا و هذا مثال على ذلك :

```
#include<iostream.h>
void main()
{
	int nwsize;
	cout<<"enter the size : \n";
	cin>>nwsize;
	int *ptr;
	ptr=new int[nwsize];
	for(int i=0;i<nwsize;i++)
	{
		*(ptr+i)=2*i;
			}
		for(int j=0;j<nwsize;j++)
	{

		cout<<*(ptr+j)<<endl;
}
}
```

الخرج

```
enter the size :
5
0
2
4
6
8
Press any key to continue_
```

عامل الموضع (&)

كما قلنا سابقاً إن المتغيرات التي نتعامل معها في لغة السي بلس بلس هي عبارة عن مواضع محجوزة في الذاكرة لذلك لا بد أن يكون لكل موضع رقم تعريفي له من قبل نظام التشغيل و هذا تابع لنظام معقد و هو نظام إدارة الذاكرة لذلك دعونا نجربن أن نعرف عن متغير و نكشف عنوانه في الذاكرة كما يلي :

```
//adress operator
#include<iostream.h>
void main()
{
	int var=5;
	cout<<"the memory adress of this
variable is :"<<&var<<endl;
}
```

الخرج

```
the memory adress of this variable is :
0x0012FF7C
Press any key to continue
```

ملاحظة قيمة الأدريس هذه قد تختلف من حاسوب لآخر حسب نظام التشغيل أو تبعاً لأمور عدة غير ذلك.

لماذا يبدأ index المصفوفات من الصفر :

هناك قاعدة ذهبية تقول :

اسم المصفوفة = عنوانها

أي أن :

```
arr=&arr
```

الشرح:
لنفترض أنه لدينا برنامج يحجز مساحة بين 1024 و 2048 من عنواين الذاكرة و يبدأ عند 1024 بمصفوفة من نوع int أي كل عنصر سيحجز له بايتين فلو قلنا في البرنامج

```
cout<<arr[0];
```

فإن الكومبايلر سيفهم لوحده أنها مؤشر لـ

```
*(1024+0)
```

فلو أردت أن تصل للعنصر الخامس يمكنك أن تكتب

```
*(1024+5)
```

وهكذا
فلو لم يتم البدء من الـindex صفر لكان أول عنصر أُهمل
و بالتالي نحن عندما نكتب

```
arr[0]
```

نحن نصرّح عن مؤشر بالمعنى العام.

و هذا كود يوضح المترادفات حول هذا المجال :

```
#include<iostream.h>
void main()
{
int arr[5]={1,2,3,4,5};
 int *ptr=arr;
for(int i=0;i<5;i++)
{
 cout<<arr[i]<<"\t";
}
 cout<<"\n\n\n";
cout<<"now !!! reading array element using pointers"<<endl;
for(int j=0;j<5;j++)
{
cout<<*(arr+j)<<"\t";
 }
 cout<<endl;
////ptr=arr=&arr
 for(int k=0;k<5;k++)
 {
 cout<<*(ptr+k)<<"\t";
}
cout<<endl;
 cout<<*arr<<endl;//=arr[0]
cout<<arr<<endl;//=&arr[0]
 cout<<&arr<<endl;//arr=&arr
}
```

التوابع (functions):

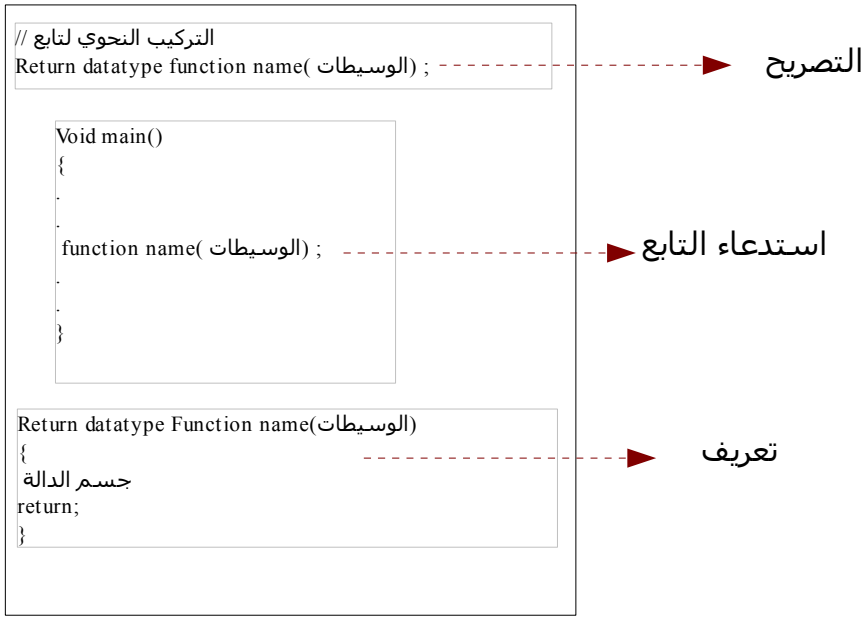
لاحظ المبرمجون و مع تطور و تعقيد البرامج أن أسلوب سرد السطور البرمجية سوف يوصلنا إلى درجة من التعقيد كبيرة لذلك اقترحوا أسلوب البرمجة البنائية و الذي يعد استخدام التوابع وجها من وجوهها فبدلاً من أن أكرر القرارات البرمجية التي أحتاج إليها في البرنامج – لنفرض ألف مرة – لماذا لا أضع هذه السطور ضمن بنية ما و أستدعي هذه البنية عند حاجتي لمجموعة القرارات هذه بدلاً من تكرارها و بالتالي سنكون استفدنا بأكثر من وجه : توفير الوقت ، تصغير عدد السطور البرمجية و بالتالي أصبح من الأسهل التعامل معها ، تسهيل موضوع صيانة البرنامج و الذي نقصد به من هذه النقطة هو:
تخيل أنك مثلاً أخطأت في كتابة هذه الأسطر البرمجية المتكررة في البرنامج و أردت تصليح هذا الخطأ فإنك و بدون البرمجة البنيوية سوف تضطر إلى تصليحه ألف مرة و لكن و باستخدام البنية هذه فإنه تصلحه مرة واحدة فقط .

التركيب النحوي :

يقسم التركيب النحوي للتوابع إلى احتمالين:

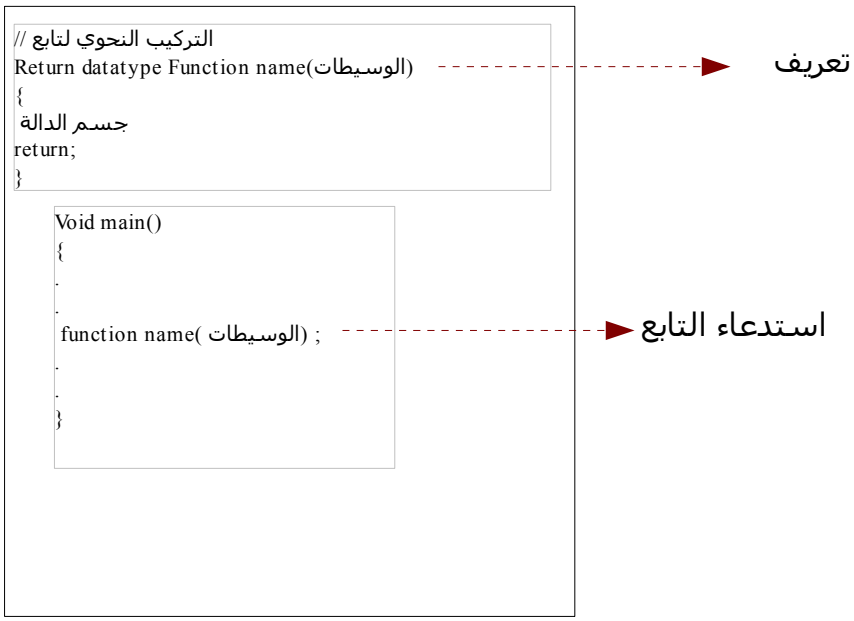
- تصريح و تعريف: و المقصود هنا أنك تصرح عن التابع قبل أن تعرفه و لفهم ذلك تابع الصورة

أدناه .



- تصريف مباشرة

أي نقوم مباشرة بتعريف التابع قبل الدالة main كما يلي :



الوسيطات و ال return datatype و الـ return :

لفهم المعنى تابع المثال التالي :
تخيل أن مجموعة القرارات البرمجية المتكررة في البرنامج هي عبارة عن حساب مجموع عددين و بالتالي كيف يمكن للتابع أن يأخذ هذين الرقمين و يجمعهما و يعيد قيمة الناتج، لذلك لابد له من وسطاء عملية الجمع و لا بد له من إعادة قيمة الناتج إلى البرنامج في مكان الاستدعاء و لهذا استخدمنا:
الوسطات(وسيطات عملية الجمع في مثالنا)،الـ return datatype ، عبارة الـ return (لإعادة القيمة و العودة إلى تطبيق الأسطر البرمجية بعد عبارة الاستدعاء)

مثال:

اكتب برنامج يطبع حرف عدد ما من المرات باستخدام التوابع.

```
#include<iostream.h>
void type(char a,int n)//تعريف التابع
{
      for(int i=0;i<n;i++)
            cout<<a<<"\n";
}
void main()
{
      char a;int b;
      cout<<"enter(char,number)"<<endl;
      cin>>a>>b;
      type(a,b);//   استدعاء التابع مع تمرير الوسيطات
}
```

الخرج

```
enter(char,number)
-
3
-
-
-
Press any key to continue
```

اكتب برنامج يطبع لك أحرف الأبجدية الإنكليزية باستخدام جدول الـ ASCII و تابع للطباعة

```
#include<iostream.h>
char typechar(char a)
{
      return a;
}
void main()
{
      char a,b;
      cout<<"enter first char and final
char\n";
      cin>>a>>b;
      for(char i=a;i<=b;i++)
cout<<typechar(i);
}
```

الخرج

```
enter first char and final char
c
g
cdefgPress any key to continue
```

تمرين :

احسب المقدار التالي باستخدام التوابع :

$$w = \begin{cases} y - z & : y.z < 0 \\ 45 & : y.z = 0 \\ z - y & : y.z > 0 \end{cases}$$

```
#include<iostream.h>
int calc(int y,int z)
{
if(y*z>0)
return z-y;
else if(y*z<0)
return y-z;
else if(y*z==0)
return 45;
}
void main()
{
      int y,z;
      cout<<"enter (y,z)\n"<<endl;
      cin>>y>>z;
cout<<calc(y,z)<<endl;
}
```

الخرج

```
enter (y,z)

3
6
3
Press any key to continue
```

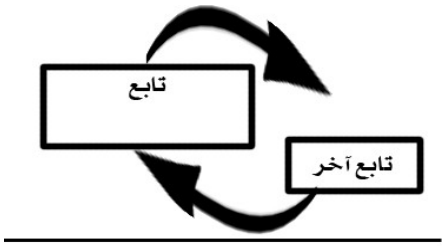
أنواع التوابع العودية:

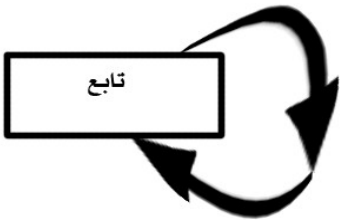
- تابع عودي مباشر:

هو الذي يستدعي نفسه أكثر من مرة.

- غير مباشر :

هو التابع الذي ستدعي غيره.





اكتب header file خاص بك:[6]

ليكن لدينا البرنامج التالي الذي يجمع أي عددين باستخدام التابع add

```
#include<iostream.h>
int add(int,int);
void main()
{
int a,b;
cout<<"enter 2 numbers ?please!\n";
cin>>a>>b;
cout<<"result:"<<add(a,b)<<endl;
}
int add(int a,int b)
{
return a+b;
}
```

ولصناعة مكتبة تحوي على تعريف بالتابع add علينا أن ننشئ الملفات التالية:

6 من كتاب learncpp الإلكتروني

add.cpp -1

```
Int add(int a,int b)
{
return a+b;
}
```

main.cpp-2

و فيه البرنامج دون استخدام تعريف للتابع .

```
#include<iostream.h>
int add(int,int);
void main()
{
int a,b;
cout<<"enter 2 numbers ?please!\n";
cin>>a>>b;
cout<<"result:"<<add(a,b)<<endl;
}
```

و إننا نستخدم التصريح عن add لكي يعلم ما الكومبايلر ما هو عمل add عن ترجمة main.cpp

add.h-3

و يتم كتابتها بالشكل التالي:

```
#ifndef ADD_H
#define ADD_H
int add(int x,int y);
#endif
```

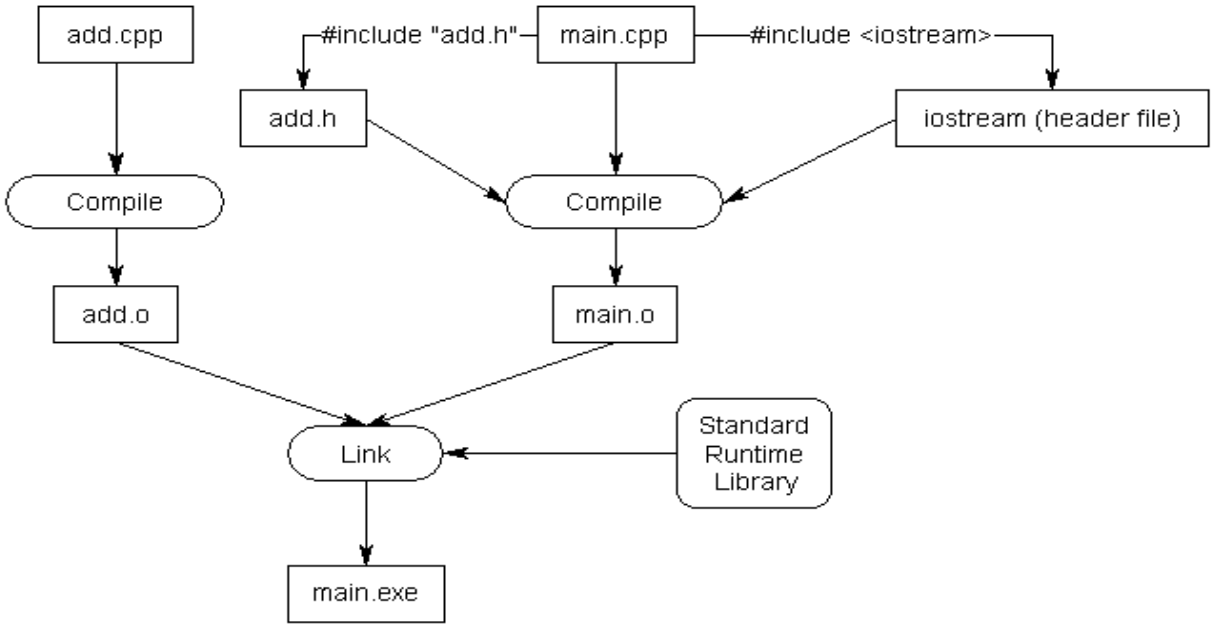
و الآن نكتب البرنامج في المترجم بعد هذه الإنشاءات

```
#include<iostream.h>
#include"add.h"
int add(int,int);
void main()
{
int a,b;
cout<<"enter 2 numbers ?please!\n";
cin>>a>>b;
cout<<"result:"<<add(a,b)<<endl;
}
```

الخرج

```
enter 2 numbers ?please!
2
3
result:5
press any key to continue.
```

و مسار العمل لإخراج البرنامج يكون في الشكل التالي :



أنت الآن مستغرب لماذا وضعنا " " للـ add و <> للـ iostream.h .

الإجابة هي أن الأقواس الزاوية توضع لتضمين مكتبة مع الكومبايلر أساساً أم استخدام إشارة الاقتباس هي لتضمين مكتبة مزودة مما يدفع للبحث عن هذه المكتبة في الـ directory الدليل الذي يوجد فيه برنامجنا أولا

مع h. أو بدونها :

إن من الأسئلة المطروحة لماذا ليس لـ iostream لاحقة h. ، والجواب لأن iostream.h يختلف عن iostream ولشرح هذا ، فإنه يتطلب عودة في تاريخ اللغة .

عندما أنشئت الـc++ كانت جميع الملفات في الـ standard runtime library تنتهي بـ h. و كانت النسخة الأصلية من cout و cin موضوعة في iostream.h و لكن عندما خضعت اللغة لمعايير ANSI[7]

واجهوا مشكلة و هي أن نقل كل التوابع من مكتبة الـ runtime إلى std namespace سيؤدي إلى عدم عمل البرامج القديمة

فلذلك قاموا بوضع مكتبات بلواحق h. ، لذلك إما أن تستخدم المكتبة عبر الـ standar و بدون لاحقة h. أو تستخدم المكتبة و ليس عبر الـ standar و لكن مع h.

7 **American National Standards Institute (ANSI)**

البنى structs

تستخدم البنى لتخزين أكثر من متغير ضمن كينونة واحدة هي البنية فمثلاً إننا نحتاج للتعريف بشخص ما على المعلومات التالية :الاسم ، العمر ، الجنس و هذا كله يندرج تحت البطاقة التعريفية للشخص و هذا يترجم برمجياً باشكل التالي:

```
Struct ID
{
int age ;
char name[256];
char sex;
};
```

للوصول إلى أي متغير في هذه البنى نستخدم الشكل التالي و ذلك بعد أن نعرف عضو من البينية

إنشاء العضو:

```
ID ahmad;
```

ألوصول إلى أي متغير من البنية في العضو :

```
ahmad.age=30;
ahmad.name[256];
ahmad.sex;
```

و الكود الكلي يصبح :

```
#include<iostream.h>
void main()
{
struct ID
{
int age ;
char name[256];
char sex;
};
ID ahmad;
ahmad.age=30;
cout<<"enter your name"<<endl;
cin>>ahmad.name;
ahmad.sex='m';
cout<<"name:"<<ahmad.name<<"\nage:"<<ahmad.age<<"
\nsex:"<<ahmad.sex<<endl;
}
```

الخرج

```
enter your name
     ahmad
   name:ahmad
     age:30
      sex:m
Press any key to continue
```

الصفوف class:

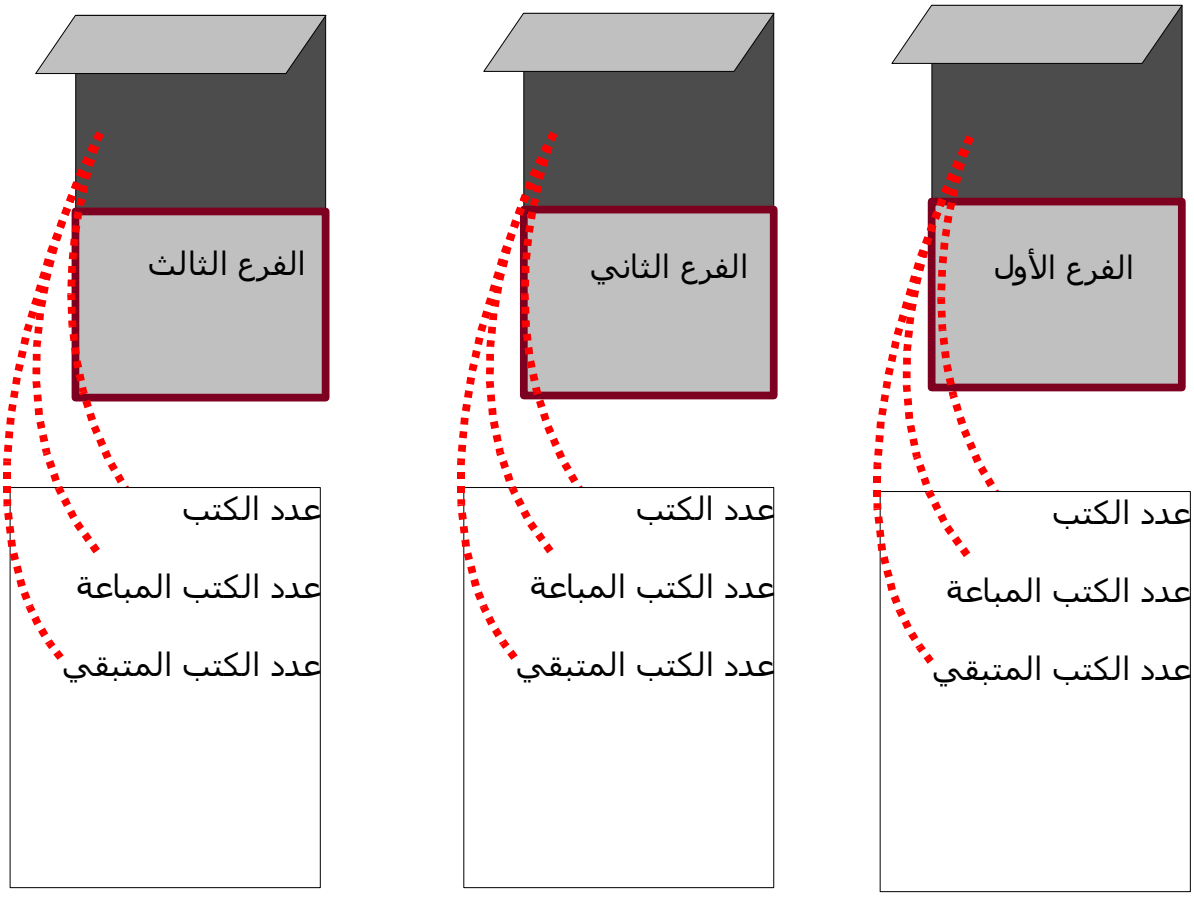
قبل البدء في شرح هذا المفهوم فإنه علينا أن نتعرف على شيء من تاريخ تطور البرمجة ، لقد بدأت البرمجة بما يسمى **بالبرمجة الإجرائية** حيث تعتمد على شرد السطور دون الاعتماد على أي تقنية بنيوية مثل التوابع مثلاً و لكن هذا المسار قد أوصل البرمجة إلى درجة كبيرة من التعقيد و الطوَل مما دفع بالمشتغلين في هذا المجال بالتفكير في حل جديد فتوصلو إلى فكرة **البرمجة البنيوية** و من سماتها أنها سببت في تفتيت الكود و تصغيره و تسهيل التعامل معه من خلال استخدام الدوال مثلاً و لكن قد وصلت إلى درجة واجهوا فيها تعقيدات مع كثرة التوابع و البنى لذلك توصلوا في النهاية إلى ماتسمى **بالبرمجة الكائنية opject oriented programming** و التي تعتمد على دمج مجموعة البيانات و الدوال التي تعمل عليها في كينونة واحد تسمى **الكائن** .

و قبل البدء في شرح الصفوف نعرج على الشرح النظري الأخير من خلال مثال يساهم في تقريب الفكرة إن شاء الله :

لنفترض أن دار من دور طباعة القرآن الكريم افتتحت عدة فروع في مدينة القدس الشريف -أعادها الله إلى عهدة السملين -لبيع المصاحف و انتشرت هذه الفروع في : حي سلوان حي الشيخ جراح و طريق الشهداء في مدينة القدس القديمة و طُلب من كل فرع البيانات التالية ليتم تزويد الفرع

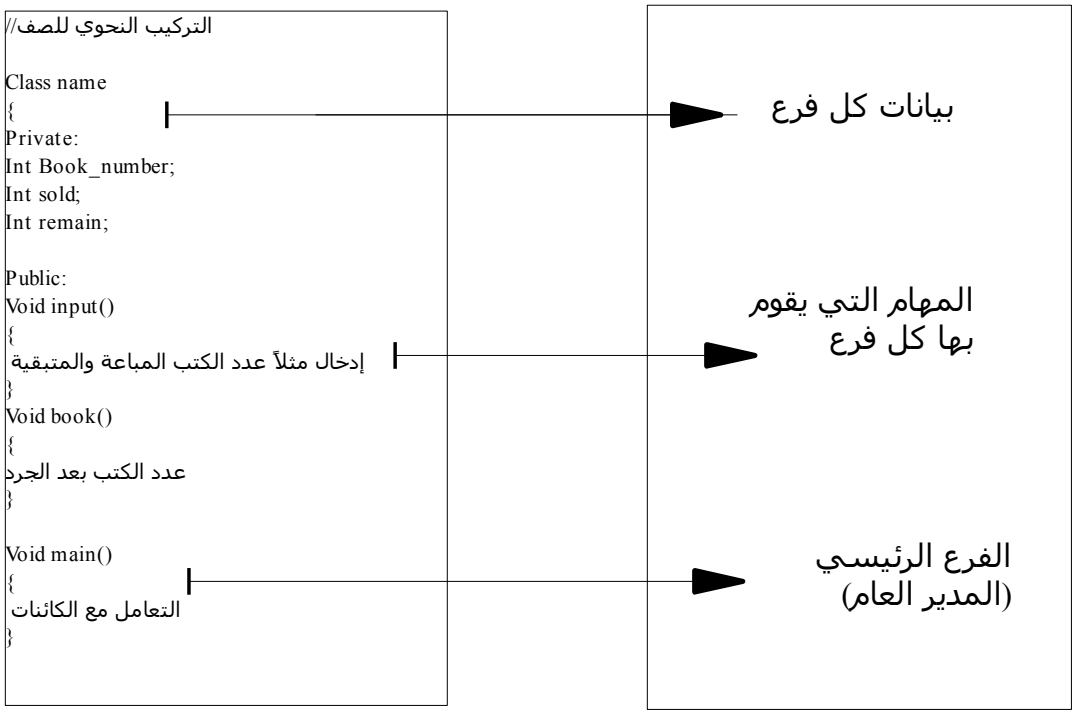
الرئيسي:

عدد الكتب -عدد الكتب المباعة اليوم – عدد الكتب المتبقية.



و بالتالي لكل فرع بياناته الخاصة و لو أردنا الحصول على أي معلومة فإننا مثلاً : نطلب من الفرع الأول أن يخبرنا بعدد الكتب و بالتالي يقوم هو يفعل الجرد و يرسل النتيجة .

و لو أردنا إسقاط هذا على واقع الكلاس فإنه يوافق مع التركيب النحوي ما يلي :



التعامل مع الكائن :

قبل التعامل مع الكائن يجب أن يتم إنشاؤه و طريقة إنشاءه تتلخص في إسباق اسم الكائن باسم الصف و بهذه العملية تكون قد حجزت في الذاكرة مكان باسم الكائن فيه مكان للتخزين يتلاءم مع نوع و عدد المتغيرات الموجود في الـprivate في الصف . وللتعامل مع هذا الكائن فإننا نطلب تنفيذ أحد الأعضاء الدالية لتتخزن هذه النتائج في تلك المساحة .
و لتوضيح فكرة إنشاء الكائن و التعامل معه راجع التمرين 35-36 في الكتاب العملي .
و هذا مثال هلى هذا العامل .

```
Void main()
{

class_name object name ; ——————▶ إنشاء كائن
object name.function member();——————▶ استدعاء عضو دالي لتنفيذه

}
```

تعريف الأعضاء الدالية خارج الصف :

إن لغة سي بلس بلس تتيح لمستخداميها أن يتم تعريف عضة دالي خارج الصف و التصريح عنه داخل الكلاس .

كما في المثال التالي:

```
#include<iostream.h>
class readtime
{
private:
        int min;
        int hor;
        int sec;
public:
        void input();
        void output();
};
void time::input()
        {
                cout<<"enter the time please\n";
                cin>>hor>>min>>sec;
        }
void time::output()
        {
                cout<<"time .now is:\n";
                cout<<hor<<":"<<min<<":"<<sec<<endl;
        }
void main()
{

time obj;
obj.input();
obj.output();
}
```

الخرج

```
enter the time please
          01
          29
          30
     time .now is:
        1:29:30
Press any key to continue
```

و يكنك أيضاً بهذه التقنية أن تصنع ملف رأسي للكلاس و بالتالي تستخدم في برنامجك أي عضو دالي دون كتبة الكلاس و ذلك بتضمين الملف الرأسي الذي يحوي الكلاس و سنطبق ذلك على المثال السابق :

1- ملف المكتبة ذو اللحقة h. :

```
//readtime.h
#ifndef READTIME_H
#define READTIME_H
class time
{
private:
int min;
int hor;
int sec;
public:
void input();
void output();
}
#endif
```

2- الملف الذي يحوي طريقة عمل الأعضاء الدالية:

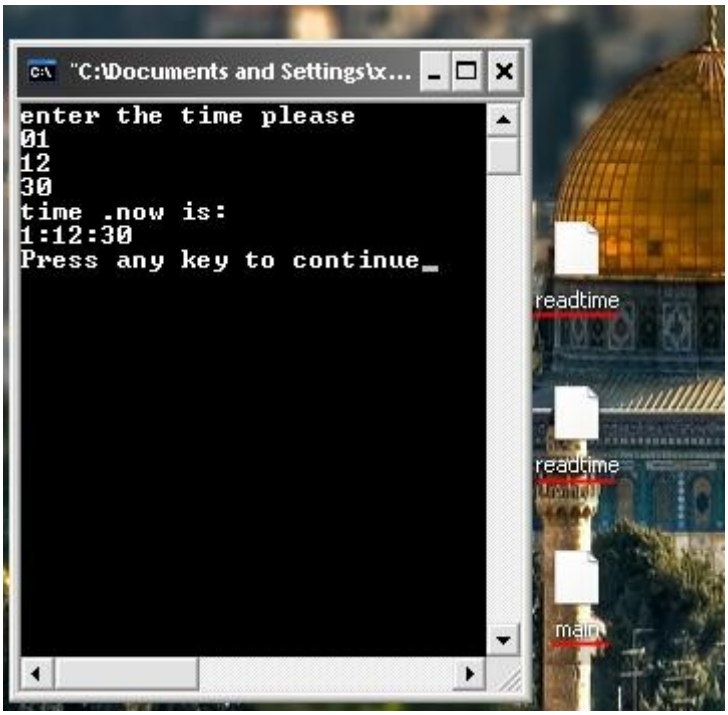
```
//readtime.cpp
#include"readtime.h"
void input()
	{
		cout<<"enter the time please\n";
		cin>>hor>>min>>sec;
	}
void output()
	{
		cout<<"time .now is:\n";
		cout<<hor<<":"<<min<<":"<<sec<<endl;
	}
```

3- الملف الرئيسي:
الذي يحوي الدالة main

```
//main.cpp
#include"readtime.h"
#include<iostream.h>
void main()
{

time obj;
obj.input();
obj.output();
}
```

و هذا مثال على التطبيق



التحديثات

إن شاء الله تعالى سيتم تحديث الكتاب كل فترة لذلك راسلنا على البريد الإلكتروني :
e7aaprj@gmail.com إلى حين أن ننشئ الموقع بإذن الله تعالى .

الحقوق

يحق لك الانتفاع بالكتاب بأي وجه مع ذكر اسم الفريق و دون أي استخدام تجاري لأن هذا الفرق وقفي و لا يجوز المتجارة باسمه بأي شكل .

الصور الموجودة من الكتاب ماكان منها من كتب خارجية فقد تم الإشارة إليه و ماعدا ذلك فهو من إنتاجيات الفريق و إن فكرة وضع الكود و بجنبه الخرج مأخوذة من كتاب :

cplusplus.comC++ Language Tutorial

تأكيد

جميع ما يكتب عبر الفريق قابل أن يحتوي على أخطاء لذا يرجى عدم اتخاذ إنتاجياتها مراجع داراسية و ذلك خوفاً من هذه المشكلة و كلنا أمل أن ننتج ما هو لا يحوي على الأخطاء العلمية .

في حال ورود خاطأ

يرجى إبلاغنا به على بريد الفريق .