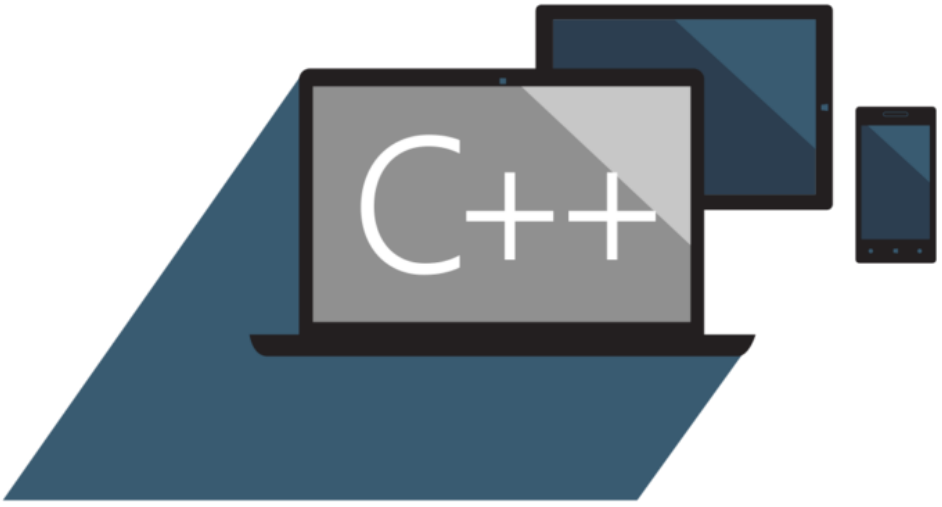
C++

The Fifth lecture



الكلية التربية

قسم الرياضيات

٣-٤ الامر for:

ويدعى ايضا بحلقة for وهو مشابه للأمر while ولكنه يحتوي على مكونان اضافيان، اذ التعبير الاول الذي يتم تقييمه مرة واحدة قبل كل شيء ومن ثم التعبير الثاني الذي يتم تقييمه مرة واحدة في نهاية كل تكرار. الصيغة العام لها هو:

```
for (expression1; expression2; expression3)
statement;
```

عندما التعبير الاول قُيم فان التعبير الثاني يقيم وينفذ في كل مرة يكون فيها الشرط صحيح والتعبير الثالث يكون للزيادة او النقصان، ان الامر for مشابه في صيغته العامة للأمر while.

```
expression1;
while (expression2) {
statement;
expression3;-
}
```

ان الاستخدام الاكثر شيوعا للحلقة for عندما يكون هنالك زيادة او نقصان للمتغير في كل مرة تكرار.

مثال (٤): جمع الارقام من ١ الى n.

```
sum = 0;
for (i = 1; i <= n; ++i)
sum += i;
```

مثال (٥): طباعة الاعداد الزوجية بين (١٠-١٠٠).

```
for (int i=10;i<=100;i+=2)
cout<<i;
```

وكما مر سابقاً يمكن حل بعض المشاكل بالطرق الثلاث

مثال (٦): المقاطع البرمجية الاتية لحساب مجموع مضاعفات العدد ٣ لغاية ٣٣:

Using for loop	Using while loop	Using do…while loop

sum = 0;
for (i = 3; i <= 33; i+=3)
sum += i;
	sum = 0; i = 3;
	while (i <= 33)
	{ sum += i; i+=3; }
		sum = 0; i = 3;
		do
		{ sum += i; i+=3; }
		while (i <= 33);

مثال (٧): اكتب برنامج لإيجاد مفكوك[١] او مضروب x.

```
#include<iostream.h>
void main()
{
int x,i,f;
cin>>x; f=1;
for(i=1;i<=x;i++)
f*=i;
cout<<"factorial x No. ="<<f;
}
```

وسيكون خزن حاصل الضرب في المتغير f وسيحمل قيمة ابتدائية بمقدار واحد بسبب ان العامل المحايد في الضرب هو واحد.

مثال (٨): برنامج لحساب قيمة p عندما $p=x^y$ (بدون استخدام دالة pow ومكتبة math.h).

```
#include<iostream.h>
void main()
{
int x,y,i,p;
cin>>x>>y; p=1;
for(i=1;i<=y;i++)
p*=x;
cout<<"power(x,y) ="<<p;
}
```

[١] اذ ان المفكوك factorial هو مضروب الارقام بدءاً من الواحد الى الرقم نفسه وبالشكل الاتي:
X!=1*2*3*…*x

بعد ادخال قيم الاساس x وقيمة الاس y وحسب معادلة استخراج القوة power سيكون التكرار للأس والضرب في الاساس وخزن الناتج في المتغير p والذي يحمل قيمة ابتدائية مقدارها واحد.

مثال (٩): برنامج لفحص قيمة n اذا كانت اولية او غير اولية.

```
#include<iostream.h>
void main()
{
int n,i,p; p=0;
cin>>n;
for(i=2;i<n;i++)
if(n%i==0) p=1;
if(p==0) cout<<"n is prime";
else cout<<"n is not prime";
}
```

ان الاعداد الاولية التي تقبل القسمة على واحد وعلى نفسها فقط، اي يجب ان نفحص ان الرقم n لا يقبل القسمة على اي من الارقام بين ٢ و1-n وبدون باقي وهكذا نعرف ان العدد اولي ومن امثلة الاعداد الاولية {3,5,7,11,…} وغيرها من الارقام.

إن ازالة كل التعابير من الحلقة for سيجعل التكرار فيها غير منتهي infinity اذ يفترض ان شرط التكرار دائما صحيح، ويكون بالصيغة الاتية:

```
for ( ; ; ) // infinite loop
something;
```

عندما تكون متغيرات الـ for متداخلة مع بعضها في العمليات غير العادية سنفصل بينها بالفارزة comma.

مثال (١٠): يوضح الفرق في التعبيرات للتداخل، كما في الشكل الاتي:

```
for (i = 0, j = 0; i + j < n; ++i, ++j)
something;
```

اذ ان المتغيرات i & j يزدادان معاً ويتوقفان عندما مجموعهما يساوي قيمة المتغير n، تتبع تنفيذ المقطع البرمجي الاتي وستشاهد المخرجات الناتجة عنه:

```
int i,j,n=6;
```

المخرجات

```
for (i = 0, j = 0; i + j < n; ++i, ++j)
cout << '(' << i << ',' << j << ")\n";
```

```
----------
(0,0)
(1,1)
(2,2)
```

مجموعة الامثلة الاتية لتوضيح أوامر التكرار الثلاثة (for وwhile وdo-while).

مثال (١١): برنامج لقراءة ١٠٠ رقم صحيح، وايجاد مجموع العناصر الزوجية ومجموع العناصر الفردية.

```
#include<iostream.h>
void main()
{
int x,i,sumo,sume;  sumo=sume=0;
for(i=1;i<=100;i++)
{cin>>x;
if(x%2==0) sume+=x;
else sumo+=x;
}
cout<<"sum of even No.="<<sume<<endl;
cout<<"sum of even No.="<<sumo;
}
```

من خلال تنفيذ البرنامج سيتم قراءة الرقم الصحيح من خلال المتغير x وفحص الرقم اذا كان زوجي من خلال استخدام علامة باقي القسمة (%) فاذا كان الرقم يقبل القسمة على ٢ بدون باقي فهذا يعني ان الرقم زوجي ويتم اضافته الى متغير خزن المجموع الزوجي sume اما اذا كان فردي سيتم اضافته الى متغير جمع الفردي sumo ومن خلال الامر for المستخدم سيتم تكرار هذه الخطوات ١٠٠ مرة.

مثال (١٢): برنامج لقراءة ٢٠٠ رقم صحيح، وحساب عدد العناصر الموجبة وعدد العناصر السالبة.

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

```
int x,i,countp,countn;  countp=countn=0;
for(i=1;i<=200;i++)
{cin>>x;
if(x>=0) countp++;
else countn++;
}
cout<<"count of positive No.="<<countp<<endl;
cout<<"count of negative No.="<<countn;
}
```

من خلال تنفيذ البرنامج سيتم قراءة الرقم الصحيح من خلال المتغير x وفحص الرقم اذا كان موجب فاذا كان الرقم اكبر او يساوي صفر فهذا يعني ان الرقم موجب ويتم زيادة العداد countp اما اذا كان سالب سيتم زيادة متغير العداد للأرقام السالبة countn ومن خلال الامر for المستخدم سيتم تكرار هذه الخطوات ٢٠٠ مرة.

مثال (١٣): برنامج لقراءة ١٥٠ رقم صحيح وحسب معدل العناصر التي تقبل القسمة على الرقمين ٤ و٥ بدون باقي.

```
#include<iostream.h>
void main()
{
int x,i,count,sum;  sum=count=0;
for(i=1;i<=150;i++)
{cin>>x;
if(x%4==0 && x%5==0) {sum+=x; count++;}
}
cout<<"Average ="<<sum/count;
}
```

لغرض حساب المعدل دائما نحتاج الى المجموع sum والعدد count وبعد فحص الرقم الصحيح x اذا كان يقبل القسمة على الرقمين ٤ و٥ بدون باقي (مثلا الرقم ٢٠) وحساب المجموع والعدد للأرقام المطابقة للشرط سيتم استخراج المعدل من خلال عبارة الطباعة مباشرة.

مثال (١٤): برنامج لقراءة ١٠٠ رقم صحيح وايجاد اكبر رقم بينهم.

```
#include<iostream.h>
void main()
{
int x,i,larg;
cin>>x; larg=x;
for(i=1;i<=99;i++)
{cin>>x;
if(x>larg) larg=x;
}
cout<<"larg No. ="<<larg;
}
```

لغرض ايجاد اكبر قيمة من ١٠٠ رقم صحيح سيتم قراءة اول رقم صحيح خارج جسم التكرار ووضعه في المتغير المؤقت larg ومن ثم قراءة باقي الارقام الـ(٩٩) وفحص اذا كان اي رقم منهم اكبر من المتغير larg سيتم تحويله الى المتغير larg وبالاتي سينتج لدينا اكبر قيمة.

ملاحظة: لغرض ايجاد اصغر قيمة سيتم قلب اشارة الشرط من اكبر الى اصغر لغرض البحث عن اصغر قيمة، اذ يكتب الشرط بالشكل الاتي:

```
If (x<small) small=x;
```

ان جميع الاسئلة او الامثلة والتي يكون في معطياتها عدد الارقام معلوماً يكون حلها باستخدام الامر for افضل واقل من عدد الخطوات البرمجية في الامرين (while او do-while) وبخلافه كل الامثلة التي يكون فيها عدد الارقام المستخدم في الاسئلة مجهولاً يفضل استخدام الامرين (while و do-while).

مثال (١٥): برنامج لحساب حاصل جمع قائمة من الارقام الصحيحة الموجبة والتي تنتهي برقم سالب.

```
#include<iostream.h>
void main()
{
int x,sum; sum=0;
```

```
cin>>x;
while(x>=0)
{
sum+=x;
cin>>x;
}
cout<<"sum to list of pos. No. ="<<sum;
}
```

بما ان قائمة الارقام غير محددة بعدد معين سيتم الاستخدام الامر while والاستمرار بإدخال الارقام الصحيحة وجمعها مازالت هي اكبر او تساوي صفر وبخلافه سيتم ايقاف التكرار وطباعة حاصل الجمع.

مثال (١٦): برنامج لقراءة قائمة من الارقام تنتهي بالرقم صفر، وحساب عدد العناصر الزوجية فيها.

```
#include<iostream.h>
void main()
{
int x,count; count=0;
cin>>x;
while(x!=0)
{
if(x%2==0)count++;
cin>>x;
}
cout<<"count of even No. ="<<count;
}
```

ان الحل باستخدام الامر while وسيستمر بادخال الارقام الصحيحة الموجبة والسالبة وحساب عدد العناصر الزوجية للارقام المدخلة مازالت لا تساوي صفر، وبما ان الشرط في هيكل while يأتي في المقدمة سيتم قراءة اول رقم في القائمة قبل الدخول الى جسم التكرار while.

مثال (١٧): برنامج لقراءة قائمة من الارقام اول رقم فيها يشابه اخر رقم، وحساب حاصل جمع تلك العناصر.

```
#include<iostream.h>
void main()
{
int x,y,sum; sum=0;
cin>>x; sum+=x;
do
{
cin>>y;
sum+=y;
} while(x!=y);
cout<<"sum of list No. ="<<sum;
}
```

تم استخدام الامر do-while لكون الشرط فيه يأتي في نهاية جسم التكرار وسندخل اول رقم x قبل الدخول في جسم التكرار وثم اضافة y التي تمثل الارقام الاتية في القائمة داخل جسم التكرار ومقارنة كل رقم داخل مع القيمة الاولى في القائمة لحين ادخال قيمة مشابه، فاذا ادخالنا الارقام الاتية 3,4,5,3 وبشكل متسلسل سيتوقف التنفيذ بعد ادخال رقم ٣ الثانية وسيطبع حاصل الجمع والذي يساوي ١٥ في هذه الحالة.

ونذكر هنا اننا يمكن تنفيذ الامثلة السابقة باستخدام الامر for من خلال استخدام اوامر اضافية مثل break وcontinue والتي سنوضحها في هذا الفصل.

٣-٥ الامر Continue:

يقوم بإيقاف التنفيذ الحالي للتكرار والقفز الى التكرار الاتي من خلال عبارة شرطية تحدد ذلك، وبهذا فان الامر continue لا يكتب او يستخدم الا داخل جسم التكرار.

مثال (١٨): قراءة قائمة من الارقام الموجبة والتي تنتهي بالرقم صفر واجراء اي معالجة على تلك الارقام سيكون الهيكل البرمجي بالشكل الاتي:

```
do {
   cin >> num;
if (num < 0) continue;
   // process num here...
} while (num != 0);
```

وهذا يكافئ الكتابة الاتية بدون استخدام:

```
continue
do {
   cin >> num;
if (num >= 0) {
   // process num here...
}
} while (num != 0);
```

واذا كانت الحلقة تدخل عدد معين من الارقام بدل ان تنتهي بالأمر صفر، قد يعبر عنها بالشكل الاتي:

```
for (i = 0; i < n; ++i) {
cin >> num;
if (num < 0) continue; // causes a jump to: ++i
// process num here...
}
```

فان اي رقم يتم ادخاله ويكون اقل من الصفر (اي سالب) سيتم القفز بالتنفيذ بواسطة الامر continue الى الحلقة التالية، اي ادخال رقم جديد.

٦-٣ الامر Break:

ان الامر break يظهر دائما داخل اوامر (for, while, do-while, switch) ويؤدي استخدامه الى الانتقال الى اول خطوة بعد نهاية جسم تلك التراكيب، ومن الخطأ استخدام الامر break خارج جسم تلك التراكيب.

مثال (١٩): قراءة قائمة من الارقام الموجبة غير محددة العدد تنتهي بالرقم صفر وباستخدام الامر for وتنفيذ اي معالجة على تلك الارقام سنستخدم خاصية for غير المنتهية لتنفيذ ذلك.

```
for( ; ; )  //infinity loop
{    cin >> num;
if (num < 0) continue;
```

```
If (num==0) break;
 // process num here...
}
```

فان التكرار غير المنتهي في تلك الحالة سيستمر بإدخال الارقام لحين تفعيل الامر break، فاذا كان الرقم المدخل سالب سيتم استخدام الامر continue لجلب التكرار الاتي وادخال رقم جديد اما اذا تم ادخال رقم صفر فسيتم ايقاف التنفيذ بواسطة الامر break كشرط للتوقف وبخلافه سيكون الرقم المدخل موجب وهو المطلوب لإجراء اي معالجة على تلك الارقام.

مثال (٢٠): برنامج لإدخال كلمة سر في عدد معين محدود من المحاولات.

```
#include<iostream.h>
void main()
{ int i,password,attempts=3;
for (i = 0; i < attempts; ++i) {
cout << "Please enter your password: ";
cin >> password;
if (password==999) // check password for correctness
break; // drop out of the loop
cout << "Incorrect!\n";
}
 if (i<3)
cout<<"correct password";
else
cout<<"Incorrect password, and  attempts terminated";
}
```

في البرنامج السابق سيسمح للمستخدم بـ٣ محاولات فقط لادخال كلمة السر بشكل صحيح، وفي حالة ادخال كلمة السر بشكل صحيح قبل ذلك سيتم الاستعانة بالامر break للخروج قبل انهاء المحاولات الثلاث.

٣-٧ الامر goto:

يستخدم الامر goto للانتقال بالتنفيذ من خطوة الى خطوة ليست ضمن تسلسل خطوات التنفيذ سواء كانت سابقة للخطوة الحالية او تأتي بعد الخطوة الحالية عندما ينطبق الشرط الذي يتم تحديده من اجل تنفيذ الانتقال.

مثال (٢١): طباعة الاعداد الزوجية من (٤-١٠٠) باستخدام الامر goto.

```
#include<iostream.h>
void main()
{
int i=4;
label:
cout<<i;
i+=2;
if (i<=100) goto label;
}
```

وهنا تم وضع نقطة الانتقال وتحديدها من خلال كتابة رمز معين مثل :label وبعد عدد من الخطوات نضع الشرط مرفق بامر الانتقال goto وفي هذا الشكل تكون الطريقة مشابه لتنفيذ الهيكل do-while.

مثال (٢٢): اعادة كتابة البرنامج السابق (بقراءة قائمة من الارقام اول رقم فيها يشبه اخر رقم وحساب حاصل جمع تلك العناصر) باستخدام الامر goto، سيكون بالشكل الاتي:

```
#include<iostream.h>
void main()
{
int x,y,sum; sum=0;
cin>>x; sum+=x;
xxx:
cin>>y;
sum+=y;
if (x!=y) goto xxx;
cout<<"sum of list No. ="<<sum;
}
```

اذ تم تحديد نقطة الانتقال من خلال الرموز :xxx وتحديد الانتقال اليها في حالة كان الرقم الاول x لا يشبه الرقم الاتي المدخل بواسطة y.

وكما ذكرنا يمكن الانتقال الى خطوات التالية وليس سابقة وحسب باستخدام الامر goto.

مثال (٢٣): طباعة الاعداد من ١ الى ١٠٠.

```
#include<iostream.h>
void main()
{
int i=1;
xxx1: cout<<i<<endl;
        i++;
if (i<=100) goto xxx1;
else goto xxx2;

xxx2: cout<<"finished";
}
```