مدخل الى الخوارزميات والبرمجة

عنوان المحاضرة: بنى التكرار
المحاضر: **د. حيدر خليل**
الكلية: الهندسة
رقم المحاضرة 7+8.

أهداف المحاضرة:
معرفة الطالب ببنى التكرار و كيفية استخدامها في المسائل البرمجية المختلفة.

- بنية التكرار while.
- بنية التكرار do while
- بنية التكرار for.

مقدمة:

الحلقات (Loops) في البرمجة ضرورية وأساسية لعدة أسباب عملية ومفاهيمية و منها:

1- تكرار تنفيذ تعليمات بنفس البنية:
   - بدلاً من كتابة نفس الكود مرارًا، تُمكّن الحلقات من تنفيذ كتلة تعليمات لعدد معين من المرّات أو حتى تحقق شرط معين.
   - مثال: طباعة الأرقام من 1 إلى 10 بدلًا من كتابة 10 عبارات طباعة.

2. التعامل مع مجموعات البيانات (قوائم، مصفوفات، مجموعات)
   - لمعالجة كل عنصر في قائمة أو جمع قيم أو البحث عن عنصر أو تطبيق تحويل.
   - مثال: جمع عناصر مصفوفة، أو تحويل كل عنصر إلى صيغة أخرى.

3. التكرار الشرطي (مرونة التحكم)
   - الحلقات مثل while أو do-while تستمر طالما تحقق شرط معين، ما يتيح تكرارًا ديناميكيًا يعتمد على بيانات وقت التشغيل (مثلاً قراءة مدخلات حتى يكتب المستخدم "خروج").

4. تقليل الأخطاء وتحسين الصيانة
   - بدلاً من تكرار نفس الكود في أماكن متعددة (وهو ما يزيد الأخطاء)، وجود حلقة مركزية يجعل التعديل أسهل وأقل عرضة للخلل.

5. تحسين الأداء والفعالية البرمجية
   - كتابة خوارزميات تعتمد على الحلقات (مثل الفرز والبحث والتجميع) يتيح تنفيذ عمليات كبيرة على بيانات كثيرة بكود منظم وواضح.
   - يُمكن دمج الحلقات مع هياكل بيانات وكائنات لتطبيق خوارزميات فعّالة.

6. أساس للهياكل المتقدمة والخوارزميات
   - الحلقات تُستخدم داخل خوارزميات متقدمة (تكرار خطوات حتى التقارب في الحوسبة العددية، أو تكرار الأعمار في الشبكات العصبية، أو المرور المتعدد على الرسوم البيانية).

7. توليد سلوك تفاعلي وزمني
   - في تطبيقات الواجهات والألعاب ومحاكاة الوقت الحقيقي، تُستخدم حلقات اللعبة (game loops) لتحديث الحالة ورسم الإطارات باستمرار.

1- البنية التكرارية while (حلقة while) :

تسمى بالبنية التكرارية ذات الشرط المسبق و تملك الصيغة العامة التالية :

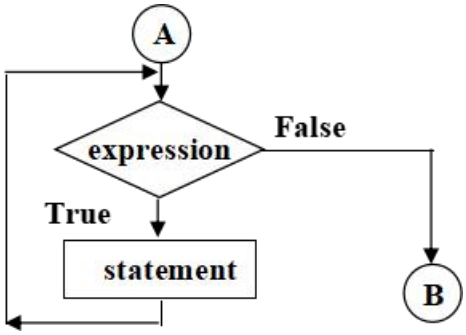
```
while ( expression ){ statement ;}
```

آلية التنفيذ:

تقدر قيمة expression فإذا كانت true يتم تنفيذ statement والتي قد تكون تعليمة مفردة أو مركبة، ثم يعاد تقدير قيمة expression مرة أخرى، فإذا كانت true تنفذ statement مرة أخرى وهكذا تتكرر هاتان الخطوتان حتى تصبح قيمة expression مساوية false، بمعنى أنه يتم تكرار تنفيذ statement طالما أن قيمة expression مساوية true وعندما تصبح تلك القيمة false يتم الخروج من البنية while.

يتم إختبار صحة الشرط قبل تنفيذ الحلقة ومن هنا أتت تسميتها بالبنية ذات الشرط المسبق. ولا بد أن يحتوي جسم الحلقة while على تعليمة تجعل الشرط غير محقق حتى يتم الخروج من هذه الحلقة أي بمعنى منع تكرار حلقة لا نهائية.

ويبين الشكل (3-4) المخطط الصندوقي للبنية while .



الشكل (1) المخطط الصندوقي للبنية while

مثال 1 :

```
#include<iostream.h>
using namespace std;
int main()
{
int x;
cout<<"enter number";
cin>>x;
while(x>5){
cout<<'*';
x--;
}
cout<<endl;
cout<<"++"<<endl;
return 0;}
```

في هذا البرنامج:

إذا تم إدخال x = 8 نجد أن شرط الحلقة while محقق(8>5) وبالتالي ستنفذ التعليمة بعد الشرط (يطبع *) ثم يتم إنقاص x بمقدار 1 فيصبح x = 7 ويعاد اختبار الشرط (7 > 5) وباعتباره محققاً يتم طباعة * وإنقاص قيمة x بمقدار 1 وهكذا حتى يصبح x = 5 فيختل الشرط ويتم الخروج من حلقة while ويكون خرج البرنامج:

```
enter number : 8
***
++
```

بينما لو تم إدخال x = 2 سيتم اختبار الشرط ويكون غير محقق وبالتالي لن تنفذ التعليمة بعد الشرط ويكون خرج البرنامج :

```
enter number : 2

++
```

مثال 2: طباعة الاعداد من 1 وحتى 100 :

```
#include <iostream>
using namespace std;
int main() {
   int i = 1;
   while (i <= 100) {
      cout << i << endl;
      ++i;
   }
   return 0;
}
```

مثال 3: طباعة مجموع الأعداد من 1 وحتى 10.

```
#include <iostream>
using namespace std;
int main() {
   int sum = 0;
   int i = 1;
   while (i <= 10) {
      sum = sum +i;
      ++i;
   }
   cout << "Sum = " << sum <<endl;
   return 0;
}
```

مثال 4: حساب مجموع عشرة أعداد مدخلة من لوحة المفاتيح.

```
#include <iostream>
using namespace std;
int main() {
   int sum = 0;
   int i = 1;
    int x;
   while (i <= 10) {
      cin>>x;
      sum = sum +x;
      ++i;
   }
   cout << "Sum = " << sum <<endl;
   return 0;
}
```

مثال 5: يطلب منك تتبع خرج البررنامج التالي:

```
#include<iostream.h>
using namespace std;
int main()
{
int pro=2;
while(pro<=60)
pro=2*pro;
cout<<pro<<endl;
return 0;
}
```

Pro Old	Pro New
2	4
4	8
8	16
16	32
32	64

في هذا البرنامج سيختل الشرط عندما يصبح pro = 64 وسيتم الخروج من while.

خرج البرنامج:

```
64
Press any key to continue
```

مثال 6 : بفرض أننا نرغب بحساب معدل عشر قيم يتم إدخالها من لوحة المفاتيح سيكون البرنامج بالشكل التالي :

```
#include <iostream>
 using namespace std;
int main()
{
int counter,grade,total,average;
total = 0;
counter = 1;
while (counter <= 10)
{
cout << "Enter grade: ";
cin >> grade;
total = total + grade;
counter = counter + 1;
}
average = total / 10;
cout << "Class average is ";
cout<< average << endl;
return 0; }
```

خرج البرنامج:

```
Enter grade: 80
Enter grade: 90
Enter grade: 96
Enter grade: 83
Enter grade: 75
Enter grade: 89
Enter grade: 79
Enter grade: 95
Enter grade: 85
Enter grade: 91
Class average is 86
```

من الضروري إعطاء المتغيرين counter,total قيم إبتدائية وإلا سيتم إعطاؤها قيم ابتدائية عشوائية أوغيرمتوقعة وبالتالي نحصل على نتائج خاطئة. المتغير counter يزيد بمقدار واحد من أجل كل عملية إدخال وعندما تصبح قيمتة 10 يختل شرط حلقة while ويتم الخروج منها.

المتغير average الذي يحسب المعدل أعلن عنه كمتغير صحيح، لذلك عند طباعتة ستعطي القسم الصحيح من المعدل المحسوب.

مثال 7: ليكن المطلوب حساب متوسط علامات الطلاب الناجحين في مقرر ما وعدد الطلاب الراسبين في ذلك المقرر. سيكون البرنامج بالشكل التالي :

```
#include <iostream>
using namespace std;
int main()
{
float average;
int counter,grade,total,count1;
total=0;
counter=0;
count1=0;
cout << "Enter grade,-1 to end";
cin >> grade;
while (grade != -1) {
if (grade >= 60){
total = total + grade;
counter = counter + 1;}
else { count1 = count1+1;}
cout << "Enter grade, -1 to end ";;
cin >> grade;
}
if (counter != 0)
{average = (float) total / counter;
cout << "Class average is " <<average << endl;
}
else
cout << "No grades were entered GE 60" << endl;
if (count1 != 0)
{cout << "the number of the failed student is " ;
cout<<count1<< endl;
}
else
cout << "No grades were entered LT 60 " << endl;
Return 0;}
```

خرج البرنامج السابق:

```
Enter grade,-1 to end :80
Enter grade,-1 to end :50
Enter grade,-1 to end :43
Enter grade,-1 to end :23
Enter grade,-1 to end :79
Enter grade,-1 to end :65
Enter grade,-1 to end :-1
Class average is 68.5
the number of the failed student is 2
```

إذا كانت جميع القيم المدخلة أصغر من 60 سيطبع البرنامج رسالة تدل على ذلك، كذلك الأمر فيما لوكانت جميع القيم المدخلة أكبر من 60.

2- البنية التكرارية do \ while

تسمى بالبنية التكرارية ذات الشرط الملحق.

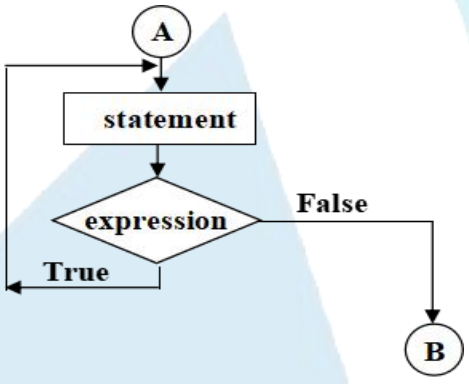
تملك الصيغة العامة التالية :

```
do statement while ( expression ) ;
```

آلية التنفيذ :

تنفذ التعليمة statement (مفردة أو مركبة) ثم يتم اختبار الشرط expression فإذا كانت نتيجة هذا الاختبار true يعاد تنفيذ التعليمة statement ومن ثم اختبار الشرط وهكذا دواليك، بمعنى أنه يتكرر تنفيذ التعليمة statement حتى يختل الشرط expression.

الشكل التالي : يبين المخطط الصندوقي للبنية do\while .



الشكل (2) المخطط الصندوقي للبنية do\while

تجدر الإشارة أنه في البنية do \ while تنفذ التعليمة statement قبل اختبار الشرط ومن هنا أتت تسميتها بالبنية ذات الشرط الملحق.

مثال 10:

أكتب برنامج يسمح بحساب معدل عشر قيم يتم إدخالها من لوحة المفاتيح معتمداً على البنية do\while.

```
#include <iostream>
using namespace std;
int main()
{
int counter,grade,total,average;
total = 0;
counter = 1;
do{
cout<<"Enter grade:";
cin>>grade;
total=total+grade;
counter = counter + 1;
}while(counter <= 10);
average = total / 10;
cout<<"Class average is "<<average<<endl;
return 0;}
```

3- البنية التكرارية for :

تسمى بالبنية ذات العدد المعروف من مرات التكرار.

تملك الصيغة العامة التالية :

for(initialization ; continuation condition ;update **) statement ;**

يتم التحكم في هذه البنية التكرارية بثلاثة أجزاء منفصلة:

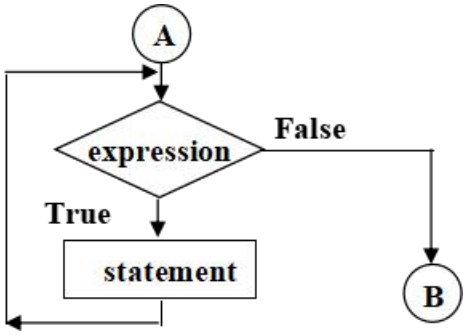
القيمة الابتدائية initialization ، شرط الاستمرار continuation condition ، القيمة الجديدة update.

آلية تنفيذ الحلقة :

يأخذ متحول الحلقة قيمة هي القيمة الابتدائية initializatio ويتم اختبار شرط استمرار الحلقة، فإن كان محققاً تنفذ التعليمة statement مفردة كانت أم مركبة، ثم يعطى متحول الحلقة قيمة جديدة ويعاد اختبار شرط استمرار الحلقة فإن كان محققاً تنفذ statement، ثم يعاد إعطاء متحول الحلقة قيمة جديدة وهكذا يستمر تنفيذ statement حتى يختل شرط استمرار الحلقة.

القيمة الابتدائية وشرط استمرار الحلقة والقيمة الجديدة يمكن أن يكونوا فارغين بدون أي قيم.

يبين الشكل (3) المخطط الصندوقي للبنية for.



الشكل (3) المخطط الصندوقي للبنية for.

جميع البرامج التي تستخدم فيها حلقة while يمكن أن تستخدم فيها حلقة for بدلاً من while فحلقتي for وwhile متكافئتان :

مثال 11 :

```
#include<iostream>
int main()
{
for(int i=1;i<3;i++)
cout<<"*";
cout<<endl;
cout<<"++"<<endl;
return 0;}
```

في هذا المقطع البرمجي يأخذ المتحول الصحيح للحلقة قيمة ابتدائية مساوية للعدد 1 وتكون نتيجة اختبار شرط استمرار الحلقة (3>1) true لذلك ستنفذ التعليمة;cout<<'*' ويتم طباعة المحرف * ، ثم تزاد بعد ذلك قيمة المتحول i وقيمة الزيادة هنا خطوة واحدة (يمكن أن تزاد بأي قيمة صحيحة) ومن ثم يعاد اختبار شرط استمرار الحلقة (3 > 2) والذي تكون قيمته true ويطبع * ، ومن ثم تزاد قيمة المتحول i فتصبح 3 وهنا يختل شرط استمرار الحلقة ويتم الخروج من for.

ويكون خرج البرنامج:

```
**
++
```

في هذا المثال القيمة الابتدائية أصغر من القيمة النهائية وتجدر الإشارة أنه يمكن أن يكون العكس.

مثال 12 :

```
#include<iostream>
int  main()
{
for(int i=2;i>0;i--)
cout<<"*";
cout<<endl;
cout<<"++"<<endl;
return 0;}
```

في هذا المثال القيمة الإبتدائية للحلقة أكبر من القيمة النهائية والعداد i يتم انقاصه حتى يصل للقيمة النهائية.

سيعطي هذا البرنامج خرجاً كالبرنامج السابق.

مثال 13:

أكتب برنامج يقوم بمايلي:

1- طباعة مجموع الأعداد المحصورة بين العدد 1 والعدد 100.
2- طباعة مجموع الأعداد الزوجية المحصورة بين العدد 1والعدد100.
3- طباعة مجموع الأعداد الزوجية التي تقبل القسمة على العدد 5 والمحصورة بين 1و100.

```
#include <iostream>
int main()
{
int sum=0,sum1=0, sum2=0,sum3=0;

cout<<"the sum of the number between 1 and 100 is:";

for(int number=1;number<=100;number +=1)
sum += number;
cout <<sum<<endl;

cout<<"the sum of the even number between 1 and 100 is:";

for(int number2=2;number2<=100; number2 +=2)
sum2 += number2;
cout<<sum2<<endl;
cout<<"the sum of the even number between 1 and 100 that divide 5 is:";

for(int number1=2;number1<=100;number1 +=2)
if(number1%5 == 0)
sum1+=number1;
cout<<sum1<<endl;
Return 0;}
```