

## مدخل إلى الخوارزميات و البرمجة

محاضرة 6 د.فادي متوج

### بنى التحكم Control Structures

#### 1. البنى الشرطية:

وقد تم دراستها في المحاضرة السابقة

#### 2. البنى التكرارية:

\* البنية التكرارية while: تم دراستها في المحاضرة السابقة.

\* البنية التكرارية do/while:

تشبه بنية التكرار do/while البنية while حيث تقوم البنية 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 الانتهاء من تنفيذ البنية do/while يتم الانتقال إلى التعليمات التي تليها مباشرة . ولها الشكل العام التالي:

```
do  
{  
    تعليمات 1  
    تعليمات 2  
};  
while (شرط);
```

#### مثال 1 :

أكتب برنامج لطباعة الأعداد من 1 - 10 بشكل عمود واحد باستخدام do/while

```
#include <iostream>  
using namespace std;  
int main()  
{  
    int i;
```

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

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

### \*البنية التكرارية for:

تدعى هذه البنية أيضا بالبنية 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 النقصان) 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 )  
{  
    تعليمة1;  
    ; تعليمة2 ;  
}
```

exp1 : يمثل تعريف وتحديد القيمة الابتدائية لعداد الحلقة .

exp2 : يمثل شرط إنهاء الحلقة أي شرط فحص النتيجة النهائية لعداد الحلقة .

exp3 : يمثل أسلوب زيادة أو إنقاص عداد الحلقة.

### مثال 1 :

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

```
#include <iostream>
using namespace std;
int main()
{
for ( int i= 1; i<=10 ;i=i+1)
cout <<i<<"\n";
return 0;
}
```

### مثال 2 :

اكتب برنامج لحساب مجموع جميع الأعداد الصحيحة من 2 إلى 100

```
#include <iostream>
using namespace std;
int main()
{
int sum = 0 ;
for ( int i = 2 ; i <= 100 ; i = i +1)
sum = sum + i ;
cout << " sum is " << sum ;
return 0;
}
```

### مثال 3 :

بفرض تم إجراء امتحان مادة ما لعشرة طلاب حيث أعطيت قائمة بأسماء الطلاب و مقابل كل اسم تم وضع القيمة 1 إذا كان الطالب ناجح و القيمة 0 إذا كان الطالب راسب في الامتحان ، اكتب برنامج يقرأ القيم المقابلة لأسماء الطلاب و يعطي عدد الطلاب الناجحين و الراسبين؟

```
#include <iostream>
using namespace std;
int 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 = p+1 ;  
    else  
        f=f+ 1 ;  
}  
cout << " passed : " <<p <<"\n" ;  
cout << " failed : " << f << "\n" ;  
return 0 ;  
}
```

```
enter result : 1  
enter result : 1  
enter result : 0  
enter result : 1  
enter result : 0  
enter result : 0  
enter result : 0  
enter result : 0  
enter result : 1  
enter result : 1  
enter result : 0  
passed : 5  
failed : 5
```

### \* بنية الاختيار المتعدد switch :

يمكن أن تصادفنا حالة خاصة في إحدى البرامج تحتوي على سلسلة من القرارات التي تتعلق بنتائج متعددة 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 بعدة اختيارات ، ولها الشكل العام التالي:

```
switch (variable name)  
{  
    case constant 1 : statement 1 ; break ;  
    case constant 2 : statement 2 ; break ;  
    case constant 3 : statement 3 ; break ;  
    .  
    .  
}
```

```
case constant n : statement n ; break ;  
default : last statement ;
```

**مثال 1:** اكتب برنامج لإعطاء اسم اليوم من أيام الأسبوع عند إعطاء رقمه

```
using namespace std;  
int main()  
{  
    int c;  
    cout << "enter number : ";  
    cin >> c;  
    switch (c)  
    {  
        case 1 : { cout << " saturday " ; break ; }  
        case 2 : { cout << " sunday " ; break ; }  
        case 3 : { cout << " monday " ; break ; }  
        case 4 : { cout << " tuesday " ; break ; }  
        case 5 : { cout << " wednesday " ; break ; }  
        case 6 : { cout << " thursday " ; break ; }  
        case 7 : { cout << " friday " ; break ; }  
        default : { cout << " that number is out of range " ; }  
    }  
    return 0;  
}
```

```
enter number : 3  
monday
```

**مثال 2:** اكتب برنامج يقوم بقراءة عددين ومن ثم يعطي ناتج جمعهما وطرحهما وضربهما لعرض ذلك شاشة خيارات

```
#include <iostream>  
using namespace std;  
int main()  
{
```

```
int n , x, y;  
cout << "Add numbers: 1 "; cout << "\n";  
cout << "Subtract numbers: 2 "; cout << "\n";  
cout << "Multiply numbers: 3 "; cout << "\n";  
cout << "*****"; cout << "\n";  
cout << "Enter the first number "; cin >> x;  
cout << "Enter the second number "; cin >> y;  
cout << "Enter the number of operation "; cin >> n;  
switch ( n )  
{  
case 1: { cout << x+y; break; }  
case 2: { cout << x-y; break; }  
case 3 : { cout << x*y; break; }  
default : { cout << "Please enter a valid option ";  
}  
}  
return 0;  
{
```

### عمليات الإسناد:

يتوفر في لغة ++C عدد من عمليات الإسناد المختصرة التي هي تعبير عن عملية الإسناد نفسها، فعلى سبيل المثال يمكن اختصار التعليمة التالية:

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

```
c+=3;
```

لتصبح بالشكل التالي:

حيث نسمي العملية += بعملية الإسناد والجمع. ويبين 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=c%10 |

### عمليات الزيادة والإنقاص بواحد :

يتوفر أيضا في لغة C++ عملية الزيادة بواحد الأحادية (++) وعملية الإنقاص بواحد الأحادية (--). 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:

ما هي نتيجة تنفيذ البرنامج التالي؟

```
#include <iostream>
using namespace std;
int main()
{
int c ;
c = 3;
cout << c << "\n" ;
cout << c ++ << "\n" ;
cout << c << "\n" ;
cout << "*****" << "\n" ;
c = 3;
cout << c << "\n" ;
cout << ++ c << "\n" ;
cout << c << "\n" ;
return 0;
```

}

```
3
3
4
*****
3
4
4
```