## المحاضرة السابعة

# التحميل الزائد للمؤثرات
# Operators Overloading

### تعريف التحميل الزائد للمؤثرات :-

هو المقدرة على ربط مؤثر معين بدالة عضو في صنف ويتم استدعاؤها عند تطبيق المؤثر على أحد كائنات الصنف .

### فائدة التحميل الزائد للمؤثرات :-

استخدام الطريقة أو الأسلوب المستخدم في التعامل مع الأنواع المعرفة مسبقاً ( الأنواع الأساسية ) في التعامل مع الأنواع المعرفة من قبل المستخدم ( الكائنات ) مثلاً : يمكن أن نقوم بتطبيق المؤثر الرياضي ( + ) [ مؤثر الجمع ] على عددين صحيحين أو حقيقيين ( أنوع أساسية ) ولكن لا يمكن أن نطبق هذا المؤثر على كائنين( أنواع معرفة ) إلا بعد تحميل المؤثر ( +) تحميلاً زائداً .

### شروط التحميل الزائد للمؤثرات :-

1. لا ينبغي تغيير وظيفة المؤثر عند التحميل مثلاً إذا قمنا بتحميل المؤثر ( + ) تحميلاً زائداً فلا ينبغي أن تستخدمه في أجراء عملية أخرى مثل ( - أو * ..... الخ ) .
2. لا ينبغي تغيير أولوية المؤثر عند التحميل .

### الصيغة العامة لدالة التحميل الزائد:-

```
Data_ type      Class_Name :: Operator (arg-List)
{
   // statements
}
```

**حيث:**

| | |
|---|---|
| **Data_ type** | نوع القيمة التي ترجعها الدالة |
| **Class-Name** | اسم الصنف التابعة له الدالة |
| **Operator** | كلمه محجوزة للإشارة إلى أن هذه الدالة دالة مؤثر |
| **arg-List** | معاملات الدالة |
| **statements** | تعليمات الدالة |

## المؤثرات التي يتم تحميلها هي :-

1. المؤثرات الأحادية Uniary Operators .

المؤثرات الأحادية هي :-

a. مؤثر التزايد ( ++) **Increment** .

b. مؤثرات التناقص (--) **Decrement** .

## مثال توضيحي للمؤثر ( ++ ) :-

```
class   alpha
{
private :
   int    a ;
public :
alpha (   )
{
int   a1  =  1 ;
a = a1;
}
void  display ( ) ;
void  operator  ++  ( ) ;
alpha (  int );
} ;
alpha :: alpha (  int a1 )
{
this  ->  a  =  a1  ;
}
void alpha  ::  display ( )
{
cout << a << "\n" ;
}
```

الآن نكتب تفاصيل الدالة **operator ++ ( )** مع ملاحظة أن الربط بين المؤثر والدالة العضو يتم باستخدام الكلمة المحجوزة **operator** مع المؤثر المراد استخدامه وهو ( ++)

```
void alpha : : operator  ++  ( )
{
this -> a =  this -> a + 1;
//  ++ this –> a ;
}
```

**تكملة البرنامج ( الدالة main ) هي :**

```
int main ( )
{
alpha  alpha1 ,  alpha2( 10 ) ;
alpha1.display ( ) ;
alpha2.display ( ) ;
++ alpha1;
++ alpha2;
// لاحظ استخدمنا المؤثر ++ مع الكائنات
// alpha2 و alpha1 لأننا قمنا بتحميله تحميلاً زائداً
alpha1.display ( ) ;
alpha2.display ( ) ;
return 0 ;
}
```

**مخرج البرنامج هو :**

**1** قيمة افتراضية للكائن الأول .

**10** قيمة الكائن الثاني .

**2** زيادة قيمة الكائن الأول بمقدار واحد .

**11** زيادة قيمة الكائن الثاني بمقدار واحد .

وبنفس الطريقة نستطيع أن نستخدم المؤثر الأحادي ( -- ) في البرنامج السابق فقط بتغيير ( ++ ) إلى ( -- ) .

ملاحظة : في المثال السابق نستطيع أن نكتب مثل العبارات التالية

```
++ alpha1 ;
++ alpha2 ;
```

ولكن لا نستطيع أن نكتب العبارة :

```
alpha1 = ++ alpha2;
```

وذلك لأن دالة التحميل الزائد ليس لها مردود لا ترجع قيمة ( **void** ) وللقيام بهذه العملية يجب إعادة تعريف الدالة لكي ترجع قيمه والقيمة هي عبارة عن كائن يتم إسنادها إلى متغير على يسار مؤثر الإسناد ( = ) .

**إذن سوف تكون دالة التحميل الزائد على الصورة التالية :**

```
alpha  alpha :: operator ++ ( )
{
 ++ this ->  a ;
return *  this ;
}
```

**لاحظ:-**

1. جعلنا الدالة ( ) **operator ++** من نوع الصنف **alpha** .
2. وضعنا ( * ) مع المؤشر **this** لأننا نريد إرجاع محتوى المؤشر **this**

**والآن نكتب البرنامج بصورة متكاملة لإرجاع كائن من دالة التحميل الزائد :**

```
#include<iostream.h>
class beta
{
private:
int b ;
public:
beta()
{
b = 1 ;
}
beta(int) ;
void display( ) ;
beta operator  ++ ( ) ;
};
beta::beta( int b1 )
{
b = b1 ;
}
beta beta :: operator ++( )
{
int b1 ;
b1 = ++ this -> b ;
return  b1 ;
}
void beta::display( )
{
cout<< b << "\n" ;
}
int main(  )
{
beta b1 , b2(20);
b1.display( );
b2.display( );
++b1 ;
b1.display( );
++b2 ;
```

```
b2.display( );
beta b3 ;
b3 = ++b1 ;
b3.display( ) ;
b3 = ++b2 ;
b3.display( ) ;
return 0;
}
```

**مخرج البرنامج هو**

```
1
20
2
21
3
22
```

**يمكن لدالة التحميل الزائد أن ترجع كائن بثلاث طرق :**

1. باستخدام المؤشر **this** كما في المثال السابق

```
alpha  alpha : : operator ++ (  )
{
++ a -> this ;
return  * this ;
}
```

2. باستخدام كائن مؤقت **Temporary Object**

```
alpha  alpha :: operator ++ (  )
{
alpha  temp ;
temp .a  = ++ this->a;
return temp ;
}
```

3. باستخدام كائن من غير اسم **Name Less Object**

```
alpha  alpha : : operator ++ (  )
{
int  a1 = ++ this->a ;
return   a 1 ;
// or
// return    alpha  ( a 1)  ;
// كأننا استدعينا دالة البناء ذات الوسيطة الواحدة
}
```

**مثال آخر : (( تحميل المؤثر ( -- ) ))**

```
#include<iostream.h>
class beta
{
int b ;
public:
beta()
{
b = 1 ;
}
beta(int) ;
void display( ) ;
beta operator  -- ( ) ;
};
beta::beta( int b1 )
{
b = b1 ;
}
beta beta :: operator --( )
{
int b1 ;
b1 = -- this -> b ;
return  b1 ;
}
void beta::display( )
{
cout<< b << "\n" ;
}
int main( )
{
beta b1 , b2(20);
b1.display( );
b2.display( );
--b1 ;
b1.display( );
b2-- ;
b2.display( );
beta b3 ;
b3 = --b1 ;
b3.display( ) ;
b3 = --b2 ;
b3.display( ) ;
return 0;
}
```

**مخرج البرنامج هو**

```
1
20
0
19
-1
18
```

## 2. تحميل المؤثرات الرياضية الثنائية :-

المؤثرات الثنائية هي :

- ✓ مؤثر الجمع( + )
- ✓ مؤثر الطرح ( - )
- ✓ مؤثر الضرب ( * )
- ✓ مؤثر القسمة ( / )
- ✓ مؤثر باقي القسمة ( % )

دائماً في تحميل المؤثرات الرياضية الثنائية لا بد من وجود قيمة راجعة ( كائن )

حيث يجب أن ترجع الدالة العضو المربوطة بالمؤثر كائن معين .

### أولاً تحميل مؤثر الجمع +

**مثال :-**

برنامج يقوم بإجراء عملية الجمع ( + ) على كائنين :

```
#include<iostream.h>
class  add
{
private:
int a  ;
public:
add( )
{
int a1 = 1 ;
a = a1 ;
}
add ( int) ;
void display ( ) ;
add operator + ( add ) ;
} ;
```

```
add :: add ( int  a1 )
{
this -> a = a1 ;
}

void add :: display( )
{
cout << a <<  "\n" ;
}

add  add :: operator + ( add p1 )
{
add temp ;
temp.a = this -> a + p1.a ;
return temp ;
}

int main ( )
{
int x ;
cout << "Enter x   :" ;
cin>> x;
add    add1 , add2( x ) ;
add1.display( );
add2.display( ) ;
add    add3 ;
add3 = add1 + add2 ;
add3.display( );
return 0 ;
}
```

**مخرج البرنامج هو**

```
1
قيمة x
x قيمة  +1
```

## مثال آخر :-

برنامج يقوم بإجراء عملية الجمع ( - ) على كائنين :

```
#include<iostream.h>
class  sub
{
int a  ;
public:
sub( )
{
int a1 = 10 ;
a = a1 ;
}
sub ( int) ;
void display ( ) ;
sub operator - ( sub ) ;
} ;
sub :: sub ( int  a1 )
{
this -> a = a1 ;
}
void sub :: display( )
{
cout << a <<  "\n" ;
}
sub  sub :: operator - ( sub p1 )
{
sub temp ;
temp.a = this -> a - p1.a ;
return temp ;
}
int main ( )
{
int x ;
cout << "Enter x   :" ;
cin>> x;
sub    sub1 , sub2( x ) ;
sub1.display( );
sub2.display( ) ;
sub    sub3 ;
sub3 = sub1 - sub2 ;
sub3.display( );
return 0 ;
}
```

**مخرج البرنامج هو**

| |
|---|
| 10<br>قيمة x<br>10 - قيمة x |

| | |
|---|---|
| ☑ | يمكن تطبيق المثال أعلاه على كافة المؤثرات الرياضية الثنائية الأخرى فقط نغير نوع المؤثر . |

## 3. تحميل مؤثر الإسناد المركب :

مؤثرات الإسناد المركب هي :

+ = , - = , * = , / =, % =

سوف نقوم بتحميل هذه المؤثرات حتى نستطيع استخدامها مع الكائنات

**مثال يوضح تحميل المؤثر =+**

صنف الأمتار :

**خصائص الصنف :-**

float meter

**العمليات ( الدوال ) :-**

- دالة بناء افتراضية ( بدون وسائط ) .
- دالة بناء ذات وسيطة واحدة .
- دالة طباعة الأمتار .
- دالة تحميل المؤثر = + .

**نص البرنامج :-**

```
#include<iostream.h>
class meters
{
private :
float m ;
public :
meters ( )
{
```

```
m = 0.0  ;
}
meters(float  m1 )
{
m = m1 ;
}
void show ( )
{
cout<< m << "\n" ;
}
meters operator += ( meters m1 )
{
//First Metods
this ->  m +=  m1.m ;
return * this ;
//Second Methods
//meters temp ;
//temp.m += m1.m ;
//return temp ;
// Third Methods
//float mm ;
//mm += m1.m ;
//return (mm);
// or
// return meter (mm) ;
}
} ;
int main ( )
{
meters m1 , m2 ( 12.5 ) ;
meters m3 ( m2 ) ;
m1.show( );
m2.show() ;
m3.show() ;
m1 += m2 ;
m1.show() ;
m2 += m3  ;
m2.show() ;
return 0 ;
}
```

**مخرج البرنامج هو**

```
0. 0
12. 5
12. 5
12. 5
25
```

| | |
|---|---|
| ☑ | يمكن تطبيق مؤثرات الإسناد المركب الأخرى<br>%= و = / و = * و = -<br>فقط نقوم بتغيير نوع المؤثر . |

## 4. تحميل مؤثرات المقارنة Over loading comparison operators

مؤثرات المقارنة هي :-

> , < , >= , <= , == , !=

| | |
|---|---|
| ☑ | مؤثرات المقارنة دائماً ترجع قيمة منطقية إذا تحقق الشرط ترجع **true** وإذا لم يتحقق ترجع **false** |

**مثال : يوضح تحميل مؤثر المقارنة أكبر من grater than ">"**

```
#include<iostream.h>
class  comp
{
private :
int     a;
public :
comp ( int n )
{
a = n ;
}
void display ( )
{
cout << a << "/n" ;
}
bool operator > ( comp c1 )
// or another method
// void operator > ( comp c1 )
```

```
{
return this -> a > c1.a ;
}
} ;
int main ( )
{
int x , y;
cout <<"Enter x :  " ;
cin >> x ;
cout <<"Enter y :  " ;
cin >> y ;
comp c1( x ) , c2( y) ;
if ( c1 > c2 )
cout << " c1 is greater " ;
else
cout << " c2 is  greater " ;
return 0 ;
}
```

| | |
|---|---|
| ☑ | يمكن تطبيق المثال أعلاه على مؤثرات المقارنة الأخرى فقط نغير نوع المؤثر . |