المحاضرة العاشرة

الوارثة المتعددة

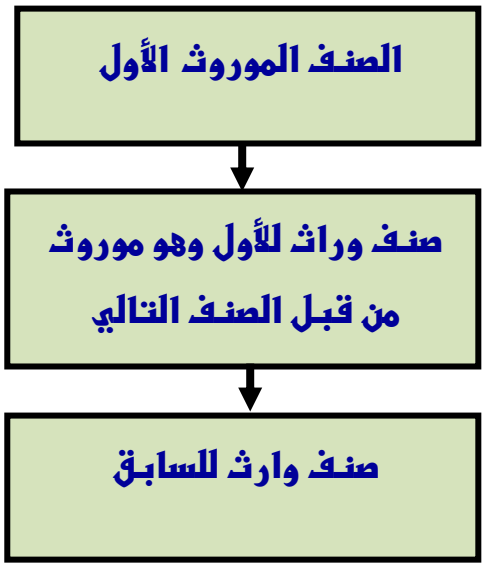
Multiple inheritance

تسمح لغة **C++** لصنف بأن يرث أكثر من صنف واحد وهذا التوارث يتحقق فقط بطريقتين :-

الطريقة الأولى

الصنف الوارث يمكن أن يستخدم كصنف موروث من قبل صنف وارث آخر .

ويمكن توضيح ذلك الهرمية التالية :-



وفي هذه الحالة الصنف الموروث الرئيسي يطلق عليه اسم "صنف موروث غير مباشر" **Indirect base class** بالنسبة للصنف الوارث الثاني .

الطريقة الثانية

الصنف الوارث يمكن أن يرث أكثر من صنف موروث

والصيغة العامة للتوارث المتعدد في هذه الحالة هي :-

```
class derived  class :name of bass class1, name of bass class 2,..name of bass class n
{
   //class body
};
```

☑	1. في حالة وراثة مجموعة من الأصناف **(base_ class)** فان دوال البناء يتم تنفيذها من اليسار إلى اليمين يترتيب التوارث أما دوال الهدم فتنفذ بطريقة عكسية . 2. في حالة هرمية التوارث فإن كل صنف وارث يجب أن يقوم بإرسال المعاملات .

مثال يوضح هرمية التوارث (توراث متعدد).

```
#include<iostream.h>
class B1
{
 int a;
public:
B1( int   x )
{
a = x;
}
int geta(    )
{
return a;
}
};
//direct inheritance for base class
class D1 : public B1
{
int b;
public:
D1(int x,int y):B1(y)
//transfer y to B1
{
b = x ;
}
int getb(     )
{
return b;
}
};
// inheritance derived class indirect class
class D2:public D1
{
int c;
public :
D2 (int x,int y,int z):D1(y,z)
//transfer parameters toD1
{
c = x;
}
//العامة والمحمية   B1  و  D1   يستطيع الوصول إلى أعضاء الصنف D2
show(  )
{
```

```
cout<<geta( )<<"\t"<<getb( )<<"\t";
cout<<c<<"\n";
}
};
int  main( )
{
int a,b,c;
cout <<"enter a and b and c \n";
cin>>a>>b>>c;
D2 ob(a,b,c);
ob.show( );
cout<<ob.geta( )<<"\t"<<ob.getb( )<<"\n";
return 0;
}
```

☑	1. الصنف **B1** يعتبر صنف موروث غير مباشر للصنف **D2** . 2. الصنف **D2** يستطيع الوصول إلي الأعضاء العامة في **B1** و **D1** . 3. كما يلاحظ في البرنامج تتم عملية إرسال المعاملات كما يلي (هرمية التوارث) **B1** ↓ **D1** ↓ **D2**

الصنف الوارث لصنفين موروثين :

مثال :-

لتوارث صنف وارث لصنفين موروثين من نوع **base class** في نفس الوقت .

```
#include<iostream.h>
class  B1
{
int a;
public:
B1 (int x)
{
a = x;
}
int geta ( )
{
 return a;
 }
```

```
 };
 // second base class
 class B2
 {
 int b;
public :
B2(int x)
{
b = x ;
}
int getb ( )
{
return b;
}
};
//direct  inheritance to two base classes
class D : public B1,public B2
{
int c;
public:
//z and y transfer by B1 and B2
D(int x, int y, int z) : B1(z) , B2(y)
{
c = x;
}
//out put function
show ( )
{
cout<<geta( )<<"\n"<<getb( )<<"\n"<<c<<"\n";
}
};
int  main( )
{
int i,j,k;
cout<<"i and j and k \n ";
cin>>i>>j>>k;
D ob(i,j,k);
ob.show( );
return 0;
}
```

ملاحظات حول البرنامج السابق :-

1. الصنف الوارث **D** يرسل إلى كل صنف موروث المعاملات

(**D** ترسل المعاملات إلى **B1** ، **B2** ولكن كلاً على حدى)

2. هرمية التوارث في هذه الحالة تكون كما يلي :

B1 (base class1) **B2 (base class2)**

D

مثال-: يوضح كيفية نداء دوال البناء ودوال الهدم عندما يكون لدينا صنفين موروثين :

```
# include<iostream.h>
class    B1
{
public :
B1 ( )
{
cout <<" Costructor B1 \n";
}
~B1 ( )
{
cout <<" Destructor B1 \n";
}
};
class B2
{
public :
B2( )
{
cout <<"Constructor B2 \n";
}
~B2( )
{
cout <<"Destructor  B2 \n";
}
};
// inheritance to base classes
class D : public B1 , public B2
{
public :
D( )
```

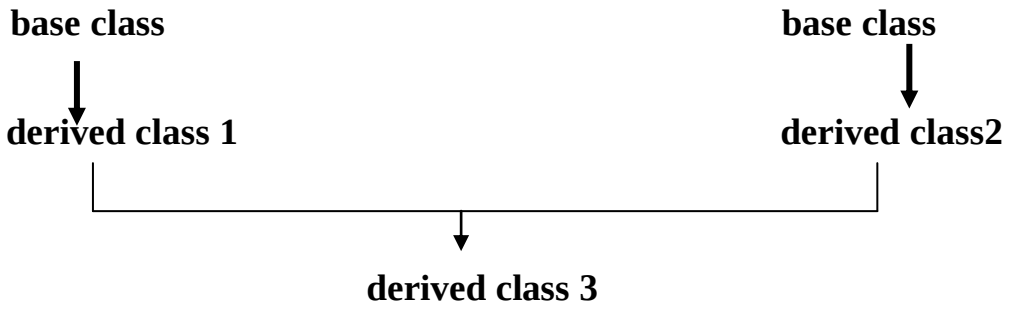
```
{
cout <<"Constructor D \n";
}
~D( )
{
cout <<"Destructor  D \n";
}
};
int  main( )
{
D  d1;
return 0;
}
```

المخرجات :

```
Constructor        B1
Constructor        B2
Constructor        D
Destructor         D
Destructor         B2
Destructor         B1
```

الصنف الموروث الوهمي Virtual Base Class

في حالة الوراثة المباشرة لأكثر من صنف قد تظهر بعض المشاكل ولفهم هذه المشاكل نناقش الهرمية التالية للتوارث :-



الصنف الموروث **base class** تتم وراثته من قبل الصنف الوراث **derived class1** و**derived class2** وبناء على هذه الهرمية فإن الصنف **base class** يتم توارثه في الصنف**derived class3** مرتين

- مره من خلال **derived class1**
- مره من خلال **derived class2**

وهذا يولد ازدواجية حيث أن الصنف **derived class3** يمتلك نسختين من الصنف **base class** مما يؤدي إلى ظهور الاستفسار التالي.

هل سيكون الإرسال إلى أعضاء الصنف الموروثه **base class** من خلال **derived class1** أو من خلال **derived class 2** .

ولهذا وحتى يتم التخلص من هذه الإزدواجية في **C++** توجد آليه خاصة تعمل فقط على إيجاد نسخة واحده من الصنف الموروث **base class** في الصنف الوارث **derived class3** ويطلق على هذه الآلية

أسم الصنف الموروث الوهمي Virtual Base Class

مثال :- على الصنف الموروث الوهمي

```
# include <iostream.h>
class    base
{
public :
 int i;
};
// inheritance base as virtual
class derived1 : virtual public base
{
public :
int j;
};
// inheritance base  class as virtual
class derived2 : virtual public base
{
public :
int k;
};
//dervied1 and derived 2  يرث الأصناف   derived3 الصنف
class derived3:public derived1,public derived2
{
public :
int product ( )
{
return (i*j*k);
}
};
int  main( )
{
derived3 ob;
cout<<"enter i and j and k \n";
cin>>ob.i>>ob.j>>ob.k;
cout<<"Result = "<<ob.product( )<<"\n";
```

```
return 0;
}
```

☑	بما أن **derived 1** و **derived 2** ترث **base** كصنف وهمي **Virtual** فإن العبارة **Ob.i ;** صحيحة ولكن أن لم يكن التوارث من نوع **Virtual** فمن الأكيد أن تظهر إزدواجية لقيمة **i** وهذا غير صحيح