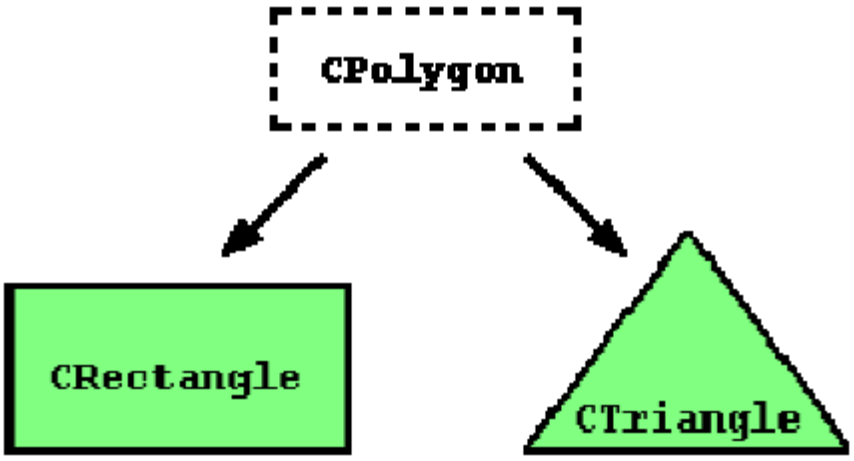
# Lecture 6

# Inheritance

A key feature of C++ classes is inheritance. Inheritance allows to create classes which are derived from other classes, so that they automatically include some of its "parent's" members, plus its own. For example, we are going. to suppose that we want to declare a series of classes that describe polygons like our CRectangle, or like CTriangle. They have certain common properties, such as both can be described by means of only two sides: height and base.

This could be represented in the world of classes with a class CPolygon from which we would derive the two other ones: CRectangle and CTriangle.



The class from which properties are inherited is called **base class** and the class to which properties are inherited is called **derived class**. Classes that are derived from others inherit all the accessible members of the base class. That means that if a base class includes a member A and we derive it to another class with another member called B, the derived class will contain both members A and B.

In order to derive a class from another, we use a colon (:) in the declaration of the derived class using the following format:

class derived_class_name: public base_class_name

{ /*...*/ };

Where derived_class_name is the name of the derived class and base_class_name is the name of the class on which it is based. The public access specifier may be replaced by any one of the other access specifiers protected and private. This access specifier describes the minimum access level for the members that are inherited from the base class.

We can summarize the different access types according to who can access them in the following way

| Access | public | protected | private |
|---|---|---|---|
| members of the same class | yes | yes | yes |
| members of derived classes | yes | yes | no |
| not members | yes | no | no |

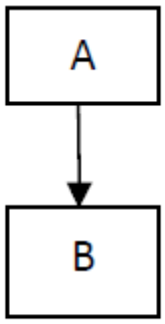
Where "not members" represent any access from outside the class, such as from main(), from another class or from a function.

Inheritance can be broadly classified into:

- Single Inheritance
- Multiple Inheritance
- Multilevel Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

## Single Inheritance

A derived class with only one base class is called single inheritance. Consider a simple example of single inheritance. In this program show a base class B and derived class D. The class B contains one private data member, one public data member, and three public member functions. The class D contains one private data members and two public member functions.




(Single inheritance)

For example :-

```
#include <iostream>
#include <conio>
class B
```

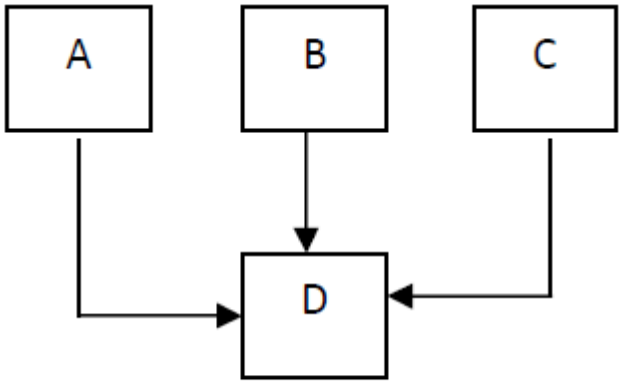
```
{
int a;
public:
int b;
void get_ab()
{
a=5;
b=10;
 }
int get_a()
{
return a;
}
void show_a()
{
cout<< "a="<<a<< "\n" ;
}
};
class D: public B
{
int c;
public:
void mul()
```

```
{
c=b*get_a();
}
void display()
{
cout<< "a="<<get_a() ;
cout<< "b="<<b ;
cout<< "c="<<c ;
}
};
main()
{
D d;
d.get_ab();
d.mul();
d.show_a();
d.display();
d.b=20;
d.mul();
d.display();
getch();
}
```

## Multiple Inheritance

A class can inherit properties from more than one class which is known as multiple inheritances. This form of inheritance can have several super classes. A class can inherit the attributes of two or more classes as shown below diagram.

Multiple inheritances allow us to combine the features of several existing classes as a starting point for defining new classes. It is like a child inheriting the physical features of one parent and the intelligent if another.




(Multiple Inheritance)

For example

```
#include <iostream>

#include <conio>

class M

{

protected:
```

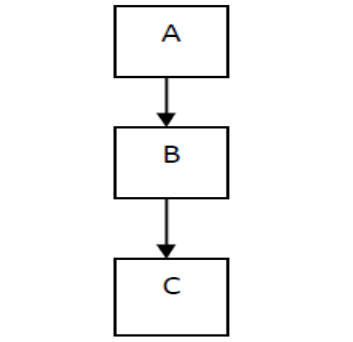
```
int m;
public :
void get_m(int);
};
class N
{
protected:
int n;
public :
void get_n(int);
};
class P :public M,public N
{
public :
void display();
};
void M :: get_m(int x)
```

```
{
m=x;
}
void N::get_n(int y)
{
n=y;
}
void P:: display()
{
cout<<"m="<<m<<"\n";
cout<<"n="<<n<<"\n";
cout<<"m*n="<<m*n<<"\n";
}
 main()
{
P p1;
p1.get_m(10);
```

```
p1.get_n(20);

p1.display();

getch();

}
```

**Multilevel Inheritance**

In multilevel inheritance class B is derived from a class A and a class C is derived from the class B.



(Multilevel inheritance)

For example

```
#include <iostream>

#include <conio>

class B

{ public :

void Display ()

{

cout << "It is class B Display" <<endl;
```

```
}
};
class D1 : public B
{
public :
void Display ()
 {
 cout << "It is class D1 Display" << endl;
 }
} ;
class D2 : public D1
{ public :
void Display () { cout << "It is class D2 Display"<< endl;
}
};
main ()
{
D2 d2 ;
d2.B::Display();
d2.D1::Display() ;
d2.Display();
getch();
}
```