Inheritance

Base-Class Access Control

Syntax –

Class derived-class: **access** base-class-name{

//Body of class

}

The object of derived class will be able to access the members of base class, on the basis of access specifier after colon.

Note:- if members of base class are inherited as public, then their specifier inside derived class will remain the same.

If members of base class are inherited as protected, then the public members of base class will be inherited protected.

If members of base class are inherited as private, then the public and protected members of base class will be inherited as private.

The private members of the base class will not be accessible by object of derived class. But **protected** specifier allows the use of base class members to the object of derived class, This is the only difference between private and protected.

See Program 16.1

Note :- There is 5 types of Inheritance i.e. Single, Multi level, Multiple, Hierarchical, Hybrid.

Note:- We have already discussed the order of constructor and destructor in case of Inheritance.

Passing Parameter to base class Constructor

To pass parameter to base class constructor from derived class object we need to modify the constructor of the derived class.

See Program 16.2

Granting Access

Consider a base class inherited as private, all public, protected members of the base class are inherited as private now.

Now we want to access some base class members with the derived class object, How do we do it?

There are two methods to get access to these members that are

1. With the help of **using** keyword

2. With the help **access declaration**

See Program 16.3

Note:- The access declaration cannot be used to lower or raise the status of the member. For e.g.:-

Class A{

Private:

Int x;

Public:

Int y;

};

Class B:private A{

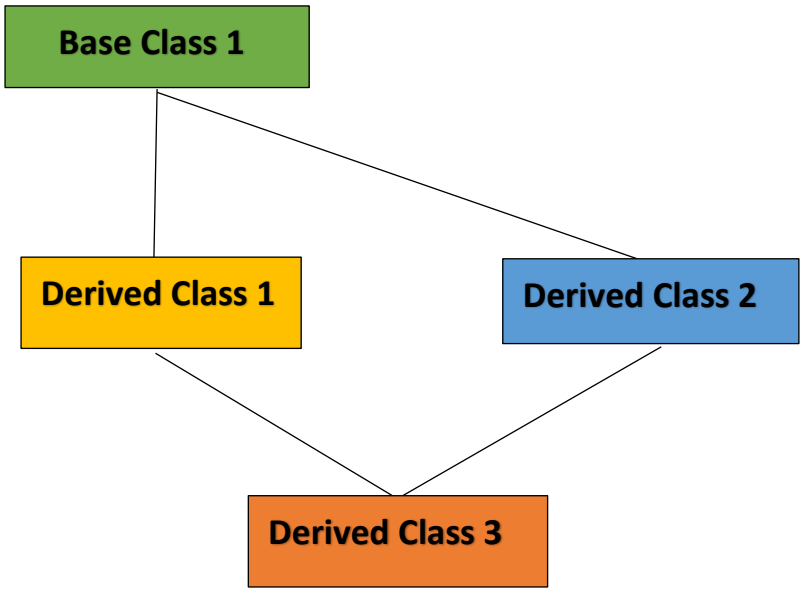
A::x=20; //This is false as the x data member is private in the class A, we cannot change it's actual status

A::y=10; //This is correct

};

Virtual Base Class

There is a small **Ambiguity** with Hybrid inheritance i.e.



Inheritance

In the above figure as we can see the **Derived Class 3** will include **Base Class 1** twice.

Hence when we try to access the member of the **Base Class 1** with the help of Object there will be Ambiguity due to presence of two Base Class.

There are two Ways of resolving the above ambiguity

1. Using Scope Resolution Operator :- base_class_name::member_to_access
2. Using virtual class

To resolve this ambiguity we use virtual class, i.e. We will inherit the base class as virtual class. This will include only one Base Class in **Derived 3 Class.**

See Program 16.4