Chapter 6 : Inheritance

Type of Inheritance with Examples

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important feature of Object Oriented Programming.

Sub Class : The class that inherits properties from another class is called Sub class or Derived Class.

Super Class : The class whose properties are inherited by sub class is called Base Class or Super class.

Why and when to use inheritance?

Consider a group of vehicles. You need to create classes for Bus, Car and Truck. The methods fuelAmount(), capacity(), applyBrakes() will be same for all of the three classes.

Implementing inheritance in C++ : For creating a sub-class which is inherited from the base class we have to follow the below syntax.

Syntax :

```
class subclass_name : access_mode base_class_name
{
  //body of subclass
};
```

Modes of Inheritance :

- **Public mode :** If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.
- **Protected mode :** If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class.
- **Private mode :** If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class.

Types of Inheritance in C++

1. <u>Single Inheritance</u> : In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is inherited by one base class only.

Syntax :

```
class subclass_name : access_mode base_class
{
  //body of subclass
};
```

2. **Multiple Inheritance :** Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e one sub class is inherited from more than one base classes.

Syntax :

```
class subclass_name : access_mode base_class1, access_mode base_class2, ....
{
  //body of subclass
};
```

3. **Multilevel Inheritance :** In this type of inheritance, a derived class is created from another derived class.

4. **Hierarchical Inheritance :** In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.

5. **Hybrid (Virtual) Inheritance :** Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.

6. **A special case of hybrid inheritance :** Multipath inheritance: A derived class with two base classes and these two base classes have one common base class is called multipath inheritance. An ambiguity can arise in this type of inheritance.

Virtual Base Classes

Virtual base classes are used in virtual inheritance in a way of preventing multiple “instances” of a given class appearing in an inheritance hierarchy when using multiple inheritances.

Abstract Base Classes

By definition, an abstract class in C++ is a class that has at least one pure virtual function (i.e., a function that has no definition). The classes inheriting the abstract class must provide a definition for the pure virtual function; otherwise, the subclass would become an abstract class itself.

Constructor in Derived Class

A constructor plays a vital role in initializing an object. An important note, while using constructors during inheritance, is that, as long as a base class constructor does not take any arguments, the derived class need not have a constructor function.

Nesting of Classes

A nested class is a class that is declared in another class. The nested class is also a member variable of the enclosing class and has the same access rights as the other members. However, the member functions of the enclosing class have no special access to the members of a nested class.

Example :

```
#include<iostream>
using namespace std;
class A {
   public :
class B {
   private:
   int num;
 public :
   void getdata(int n) {
```

```
      num = n;
       }
       void putdata() {
          cout<<"The number is "<<num;
       }
   };
};
int main() {
   cout<<"Nested classes in C++"<< endl;
   A :: B obj;
```

```
    obj.getdata(9);
    obj.putdata();
    return 0;
}
```

Output :

```
Nested classes in C++
The number is 9
```