C++ Inheritance

One of the most important concepts in object-oriented programming is that of inheritance. Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and fast implementation time.

When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the **base** class, and the new class is referred to as the **derived** class.

The idea of inheritance implements the **is a** relationship. For example, mammal IS-A animal, dog IS-A mammal hence dog IS-A animal as well and so on.

Base and Derived Classes

A class can be derived from more than one classes, which means it can inherit data and functions from multiple base classes. To define a derived class, we use a class derivation list to specify the base class(es). A class derivation list names one or more base classes and has the form −

```
class derived-class: access-specifier base-class
```

Where access-specifier is one of **public, protected,** or **private**, and base-class is the name of a previously defined class. If the access-specifier is not used, then it is private by default.

Consider a base class **Shape** and its derived class **Rectangle** as follows −

```cpp
#include <iostream>

using namespace std;

// Base class
class Shape {
   public:
      void setWidth(int w) {
         width = w;
      }
      void setHeight(int h) {
         height = h;
      }
```

Live Demo

```cpp
    protected:
        int width;
        int height;
};

// Derived class
class Rectangle: public Shape {
    public:
        int getArea() {
            return (width * height);
        }
};

int main(void) {
    Rectangle Rect;

    Rect.setWidth(5);
    Rect.setHeight(7);

    // Print the area of the object.
    cout << "Total area: " << Rect.getArea() << endl;

    return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
Total area: 35
```

Access Control and Inheritance

A derived class can access all the non-private members of its base class. Thus base-class members that should not be accessible to the member functions of derived classes should be declared private in the base class.

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

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

A derived class inherits all base class methods with the following exceptions −

•   Constructors, destructors and copy constructors of the base class.

•   Overloaded operators of the base class.

•   The friend functions of the base class.

Type of Inheritance

When deriving a class from a base class, the base class may be inherited through **public, protected** or **private** inheritance. The type of inheritance is specified by the access-specifier as explained above.

We hardly use **protected** or **private** inheritance, but **public** inheritance is commonly used. While using different type of inheritance, following rules are applied −

•   **Public Inheritance** − When deriving a class from a **public** base class, **public** members of the base class become **public** members of the derived class and **protected** members of the base class become **protected** members of the derived class. A base class's **private** members are never accessible directly from a derived class, but can be accessed through calls to the **public** and **protected** members of the base class.

•   **Protected Inheritance** − When deriving from a **protected** base class, **public** and **protected** members of the base class become **protected** members of the derived class.

•   **Private Inheritance** − When deriving from a **private** base class, **public** and **protected** members of the base class become **private** members of the derived class.

Multiple Inheritance

A C++ class can inherit members from more than one class and here is the extended syntax −

```
class derived-class: access baseA, access baseB....
```

Where access is one of **public, protected,** or **private** and would be given for every base class and they will be separated by comma as shown above. Let us try the following example −

Live Demo

```cpp
#include <iostream>

using namespace std;

// Base class Shape
class Shape {
    public:
        void setWidth(int w) {
            width = w;
        }
        void setHeight(int h) {
            height = h;
        }

    protected:
        int width;
        int height;
};

// Base class PaintCost
class PaintCost {
    public:
        int getCost(int area) {
            return area * 70;
        }
};

// Derived class
class Rectangle: public Shape, public PaintCost {
    public:
        int getArea() {
            return (width * height);
        }
};

int main(void) {
    Rectangle Rect;
    int area;

    Rect.setWidth(5);
    Rect.setHeight(7);

    area = Rect.getArea();
```

```
      // Print the area of the object.
      cout << "Total area: " << Rect.getArea() << endl;

      // Print the total cost of painting
      cout << "Total paint cost: $" << Rect.getCost(area) << endl;

      return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
Total area: 35
Total paint cost: $2450
```