

Class- Inheritance

Introduction

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.

- This existing class is called the base class, and the new class is referred to as the derived class.
- The new class should inherit the members of an existing class.
- The idea of inheritance implements the is a relationship.

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.

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.

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.

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....
```

Example

```
// derived classes
#include <iostream>
using namespace std;

class Polygon {
protected:
    int width, height;
public:
    void set_values (int a, int b) { width=a; height=b; }
};

class Rectangle: public Polygon {
```

```
    public:
        int area ( void ) { return width * height; }
};

class Triangle: public Polygon {
    public:
        int area ( void ) { return width * height / 2; }
};

int main () {
    Rectangle rect;
    Triangle trgl;
    rect.set_values (4,5);
    trgl.set_values (4,5);
    cout << rect.area() << '\n';
    cout << trgl.area() << '\n';
    return 0;
}
```