C++ Inheritance

In C++, inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such way, you can reuse, extend or modify the attributes and behaviors which are defined in other class.

In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

Advantage of C++ Inheritance

Code reusability: Now you can reuse the members of your parent class. So, there is no need to define the member again. So less code is required in the class.

C++ Single Level Inheritance Example: Inheriting Fields

When one class inherits another class, it is known as single level inheritance. Let's see the example of single level inheritance which inherits the fields only.

```
1. #include <iostream>
2. using namespace std;
3.  class Account {
4.    public:
5.    float salary = 60000;
6.  };
7.    class Programmer: public Account {
8.    public:
9.    float bonus = 5000;
10.  };
11.int main(void) {
12.    Programmer p1;
13.    cout<<"Salary: "<<p1.salary<<endl;
14.    cout<<"Bonus: "<<p1.bonus<<endl;
15.   return 0;
16.}
```

Output:

```
Salary: 60000
Bonus: 5000
```

Syntax Of Inherit Base Class

```
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:

```
#include <iostream>
 using namespace std;
// Base class
class Shape  {
  public:
    void setWidth(int w) {
      width = w;
    }
    void setHeight(int h) {
      height = h;
    }
  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;
}
```

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

Advantage of inheritance

If we develop any application using this concept than that application have following advantages,

- Application development time is less.
- Application take less memory.
- Application execution time is less.
- Application performance is enhance (improved).
- Redundancy (repetition) of the code is reduced or minimized so that we get consistence results and less storage cost.

Types of Inheritance

Based on number of ways inheriting the feature of base class into derived class it have five types they are:

- Single inheritance
- Multiple inheritance
- Hierarchical inheritance
- Hybrid inheritance

Single inheritance

Multiple Inheritances

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 2: Multiple Inheritance in C++ Programming

This program calculates the area and perimeter of an rectangle but, to perform this program, multiple inheritance is used.

```
#include <iostream>
using namespace std;

class Mammal {
  public:
    Mammal()
    {
      cout << "Mammals can give direct birth." << endl;
    }
};

class WingedAnimal {
  public:
    WingedAnimal()
    {
      cout << "Winged animal can flap." << endl;
    }
};

class Bat: public Mammal, public WingedAnimal {
```

```
};

int main()
{
   Bat b1;
   return 0;
}
```

Output

```
Mammals can give direct birth.

Winged animal can flap.
```

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:

```
#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
```

Hierarchical Inheritance

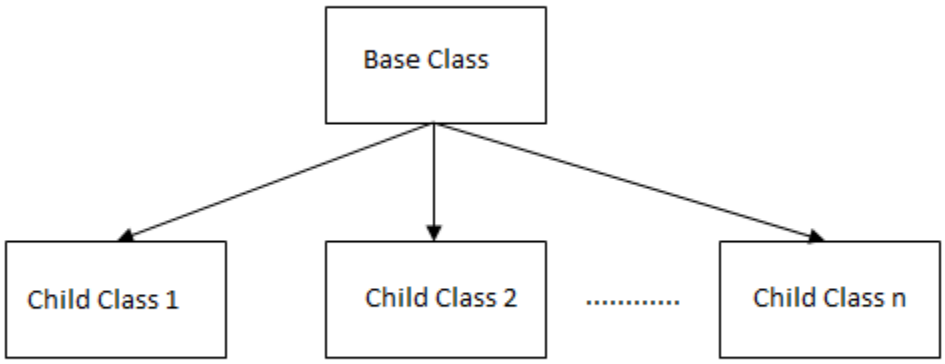

Fig: Hierarchical Inheritance

Syntax of Hierarchical Inheritance

```
class base_class {

     ... .. ...

}

class first_derived_class: public base_class {

     ... .. ...

}

class second_derived_class: public base_class {

     ... .. ...

}

class third_derived_class: public base_class {

     ... .. ...

}
```

Hierarchical inheritance to get square and cube of a number program in C++.

```
/*C++ program to demonstrate example of hierarchical inheritance to get
square and cube of a number.*/
#include <iostream>
using namespace std;

class Number
{
	private:
		int num;
	public:
		void getNumber(void)
		{
			cout << "Enter an integer number: ";
			cin  >> num;
		}
		//to return num
		int returnNumber(void)
		{ return num; }
};

//Base Class 1, to calculate square of a number
class Square:public Number
{
	public:
	int getSquare(void)
	{
		int num,sqr;
		num=returnNumber();   //get number from class Number
		sqr=num*num;
		return sqr;
	}
};

//Base Class 2, to calculate cube of a number
class Cube:public Number
{
	private:

	public:
	int getCube(void)
	{
		int num,cube;
		num=returnNumber();   //get number from class Number
```

```
                cube=num*num*num;

                return cube;
        }
};
int main()
{
                Square objS;
                Cube objC;
                int sqr,cube;

                objS.getNumber();
                sqr =objS.getSquare();
                cout << "Square of "<< objS.returnNumber() << " is: " << sqr  <<
endl;

                objC.getNumber();
                cube=objC.getCube();
                cout << "Cube   of "<< objS.returnNumber() << " is: " << cube <<
endl;

                return 0;
}
```

```
Enter an integer number: 10

Square of 10 is: 100

Enter an integer number: 20
```

Cube of 10 is: 8000

Multilevel Inheritance

In C++ programming, not only you can derive a class from the base class but you can also derive a class from the derived class. This form of inheritance is known as multilevel inheritance.

```
class A

{

... .. ...

};

class B: public A

{

... .. ...

};

class C: public B

{

... ... ...

};
```

Here, class B is derived from the base class A and the class C is derived from the derived class B.

Example 1: C++ Multilevel Inheritance

```
#include <iostream>
using namespace std;

class A
{
    public:
      void display()
      {
          cout<<"Base class content.";
      }
};

class B : public A
{

};

class C : public B
{

};

int main()
{
    C obj;
    obj.display();
    return 0;
}
```

Output

```
Base class content.
```