# BCA SEMESTER-II

# Object Oriented Programming using C++

# Paper Code: BCA CC203

# Unit :4

# Types of Inheritance in C++


## By: Ms. Nimisha Manan

**Assistant Professor**

**Dept. of Computer Science**

**Patna Women's College**

# Types of Inheritance in C++

1) Single inheritance

2) Multilevel inheritance

3) Multiple inheritance

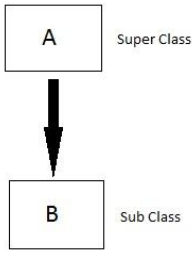
4) Hierarchical inheritance

5) Hybrid inheritance

## 1) Single inheritance

In Single inheritance one derived class inherits from one base class only . It is the most simplest form of Inheritance.

For example: Lets say we have class A and B




### Example of Single inheritance:

```
//Class B inherits from Class A
#include <iostream.h>
class A
{ public:
     A()
     {cout<<"Constructor of A class"<<endl; }
};
class B: public A
{  public:
     B()
     {cout<<"Constructor of B class";}
};
int main()
{
   //Creating object of class B
   B obj;
   return 0;
}
```

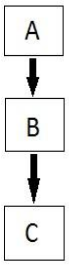
Output:

```
Constructor of A class
Constructor of B class
```

## 2)Multilevel Inheritance

In this type of inheritance a  class can be derived  from another derived class. The Super class for one, is sub class for the other.

### Example of Multilevel inheritance:




```
//class C inherits from class B and class B inherits from  class A
#include <iostream.h>
class A
{ public:
      A()
      {cout<<"Constructor of A class"<<endl; }
};

class B: public A
{ public:
  B()
   { cout<<"Constructor of B class"<<endl; }
 };

class C: public B
{ public:
    C()
      { cout<<"Constructor of C class"<<endl; }
};
int main() {
  //Creating object of class C
  C obj;
  return 0;
}
```

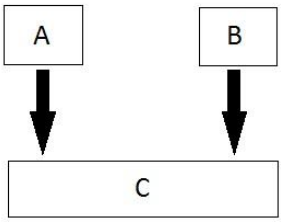
Output:

```
Constructor of A class
Constructor of B class
Constructor of C class
```

## 3) Multiple Inheritance

In multiple inheritance, a derived class can inherit from more than one Base classes. It means that Multiple inheritance allow a single child class to have multiple parent classes.

### Example of Multiple Inheritance:




```
// Class C inherits Class A and Class B both
#include <iostream.h>
class A
{ public:
      A()
      { cout<<"Constructor of A class"<<endl; }
};

class B
{ public:
      B()
      { cout<<"Constructor of B class"<<endl; }
};

class C: public A, public B
{ public:
       C()
        { cout<<"Constructor of C class"<<endl; }
};

int main()
{  //Creating object of class C
   C obj;
   return 0;
}
```

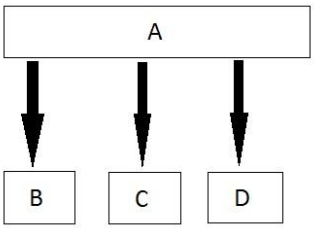
Output:

```
Constructor of A class
Constructor of B class
Constructor of C class
```

## 4)Hierarchical Inheritance

In this type of inheritance, a single Base class is inherited by multiple derived classes.

### Example of Hierarchical inheritance:



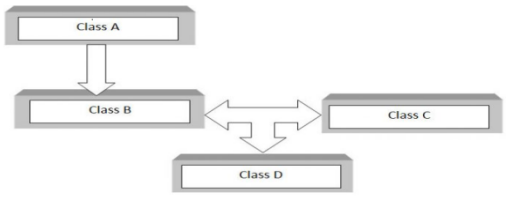

```
//Class B and Class C inherit from  class A
#include <iostream.h>
class A
{
  public:
    A()
     { cout<<"Constructor of A class"<<endl;}
  };

class B: public A
{
  public:
     B()
     { cout<<"Constructor of B class"<<endl; }
};
class C: public A
{
   public:
     C()
      { cout<<"Constructor of C class"<<endl; }
};
class D: public A
{
   public:
     D()
      { cout<<"Constructor of D class"<<endl; }
};
int main()
 {
   C obj;        //Creating object of class C
   return 0;
}
```

Output:

```
Constructor of A class
Constructor of C class
```

## 5) Hybrid Inheritance

Hybrid inheritance is a combination of more than one type of inheritance. For example, A child and parent class relationship that follows multiple and hierarchical inheritance both can be called hybrid inheritance.




**Example of Hybrid inheritance :**

```
#include <iostream.h>

class A

{       public:

        int x;

};

class B : public A

{       public:

        B()      //constructor to initialize x in base class A

        { x = 10; }

};

class C

 {      public:

        int y;

        C()   //constructor to initialize y

        { y = 4; }

};

class D : public B, public C   //D is derived from class B and class C

{       public:

        void sum()

        { cout << "Sum= " << x + y;}

};

int main()
         { D obj1;          //object of derived class D
           obj1.sum();   return 0; }
```