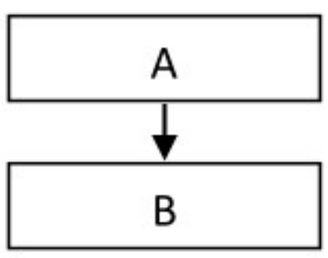
UNIT-4 - INHERITANCE

1 Explain Concept of Inheritance OR Explain type of inheritance with example.

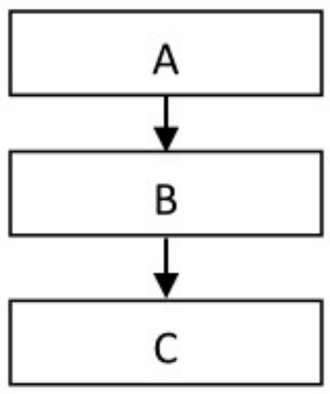
- Inheritance એક એવી process છે કે જેની મદદથી એક class એ બીજા class ની properties અને methods ને મેળવી શકે છે
- જુના class ઉપરથી નવો class બનાવવાની process ને inheritance કહેવામા આવે છે.
- નવા class ને derived class કહેવામાં આવે છે અને જુના class ને base class કહેવામાં આવે છે.
- derived class માં base class ના બધા features હોઈ છે અને પ્રોગ્રામર derived class માં નવા features ઉમેરી પણ શકે છે.
- **Type of inheritance :**

Single Inheritance



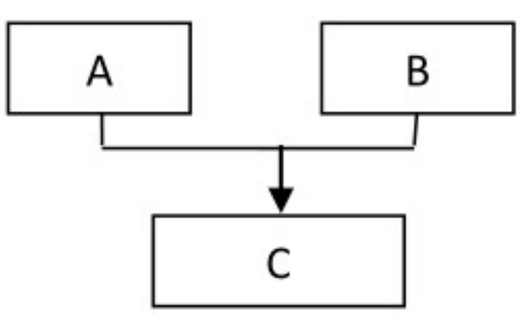
- A class જો કોઈ single class માંથી derive થયો હોય તો તેને single inheritance કહેવામાં આવે છે.
- B Class એ *A* class માંથી derive થયેલો છે

Multilevel Inheritance



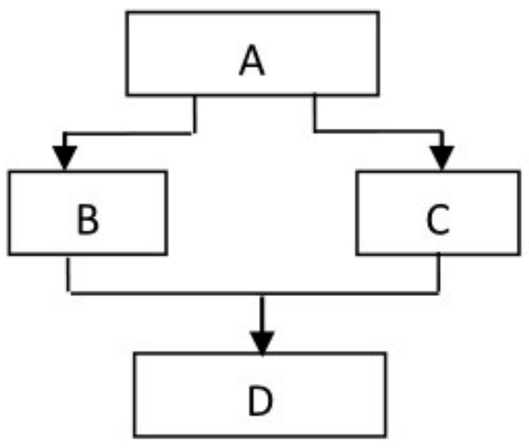
- એક Class કે જે એક અલગ class માંથી derive કરવામાં આવ્યો હોય અને તે અલગ class બીજા અલગ class માથી derive કરવામાં આવ્યો હોય તો તેને multilevel inheritance કહેવામાં આવે છે
- અહીં, *C* class એ *B* class માંથી derived થયેલો છે અને *B* class એ *A* class માંથી derived થયેલો છે, તેથી તેને multilevel inheritance કહેવામાં આવે છે.

Multiple Inheritance



- જો કોઈ એક class એક કરતા વધારે base class માંથી derived થયેલો હોય તો તેને multiple inheritance કહેવામાં આવે છે.
- અહીં, *C* class એ class *A* અને class *B* બંને માંથી derived થયેલો છે

Hybrid Inheritance



- તે ઉપરના કોઈપણ inheritance ના type નું combination છે. તે કા તો multiple અથવા multilevel અથવા કોઈપણ નું combination છે.
- અહીં, class *B* અને class *C* એ class *A* માંથી derive કરવામાં આવ્યો છે અને class *D* એ class *B* અને class *C* માંથી derive કરવામાં આવ્યો છે
- Class *A*, class *B* અને class *C* એ Hierarchical Inheritance નું example છે. અને class *B*, class *C* અને class *D* એ Multiple Inheritance નું example

છે તેથી આ hybrid inheritance એ Hierarchical અને Multiple Inheritance નું combination છે.

2 Define Derived Class in inheritance.

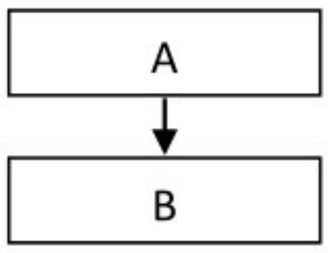
- derived class તેની પોતાની details અને base class સાથેના relationship ને specify કરીને બનાવવા માં કરવામાં આવે છે.
- derived class નું General form નીચે બતાવ્યા પ્રમાણે છે:

```
Class derived-class-name : visibility-mode  base-class-mode
{
.....// members of derived class
.....//
}
```

- જ્યાં class એ જરૂરી કીવર્ડ છે, derived-class-name એ derived class ને આપેલ નામ છે, base-class-name નું નામ base class નું નામ છે અને ':' (કોલોન) સૂચવે છે કે derived-class-name છે એ base-class-name પરથી derived કરવામાં આવ્યું છે, વિઝિબિલીટી-મોડ optional છે અને જો રાખ્યું હોય તો તે private અથવા public હોઈ શકે છે. default વિઝિબિલીટી-મોડ private છે. Visibility mode એ દર્શાવે છે કે base class ના features privately લેવામાં આવ્યા છે કે publicly લેવામાં આવ્યા છે.

3 Explain Single Level Inheritance with example.

- એવું Inheritance કે જેમાં derived class એ ખાલી એક જ base class પર થી derive કરવામાં આવ્યો હોય તેને **single level inheritance** કહેવામા આવે છે.



- **Syntax :**

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

- **Example:**

```
#include<iostream>
using namespace std;
class A
{
      public:
            void methodA()
            {
                  cout<<"Hello, I am method of class A"<<endl;
            }
};
```

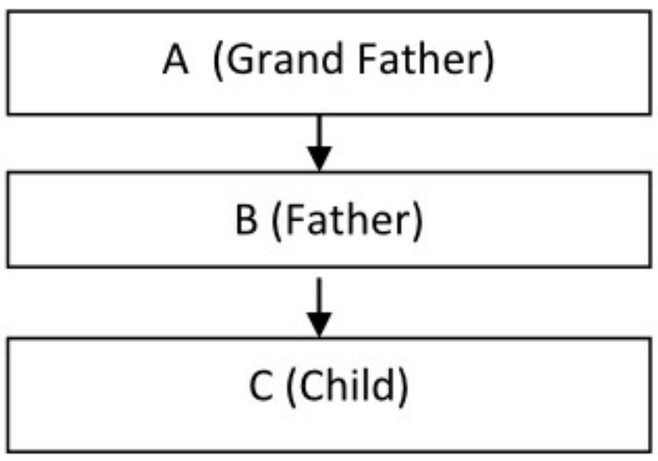
```
class B : public A
{
      public:
            void methodB()
            {
          cout<<"Hello, I am method of class B"<<endl;
            }
};
int main()
{
      B obj2;
      obj2.methodA();
      obj2.methodB();
      return 0;
}
```

Output:

"Hello, I am method of class A"
"Hello, I am method of class B"

4 Explain Multilevel Inheritance with example.

- આ type ના inheritance માં derived class એ બીજા derived class માથી બનાવવામાં આવે છે.
- જો class બીજા derived class માથી derive થયેલો હોય તો તેને multilevel inheritance કહેવામા આવે છે. તેથી C++ ના multilevel inheritance માં class ને એક કરતા વધુ parent class હોય છે.




```
class A                    //Base class
{
        .....
};
class B : pullic A         //B derived from A
{
        .....
};
class C : public B         //C derived from B
{
        .....
};
```

Example :

```
#include<iostream>
```

```
using namespace std;
class A
{
        public:
                void m_1()
                {
                cout<<"Hello, I am method of Class A"<<endl;
                }
};
class B : public A
{
        public:
                void m_2()
                {
                cout<<"Hello, I am method of Class B"<<endl;
                }
};
class C : public B
{
        public:
                void m_3()
                {
                cout<<"Hello, I am method of Class C"<<endl;
                }
};
int main()
{
        C obj;
        obj.m_1();
        obj.m_2();
        obj.m_3();
        return 0;
}
```

Output :

"Hello, I am method of Class A"
"Hello, I am method of Class B"
"Hello, I am method of Class C"

Example 2 :

```
#include <iostream>
using namespace std;
class base //single base class
{
        public:
        int x;
        void getdata()
        {
        cout << "Enter value of x= "; cin >> x;
```

```
        }
};
class derive1 : public base // derived class from base class
{
        public:
        int y;
        void readdata()
        {
           cout << "\nEnter value of y= "; cin >> y;
        }
};
class derive2 : public derive1   // derived from class derive1
{
        private:
        int z;
        public:
        void indata()
        {
        cout << "\nEnter value of z= "; cin >> z;
        }
        void product()
        {
           cout << "\nProduct= " << x * y * z;
        }
};
int main()
{
    derive2 a;     //object of derived class
    a.getdata();
    a.readdata();
    a.indata();
    a.product();
    return 0;
}
```

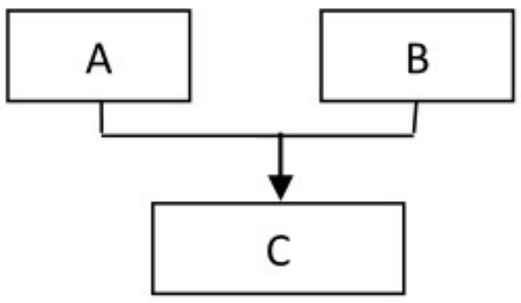
Output :

```
Enter value of x=2
Enter value of y=2
Enter value of z=2
Product=8
```

5 Explain Multiple Inheritance with example.

- જો કોઈ class બે કે તેથી વધુ base classes માંથી derived કરવામાં આવ્યો હોય તો તેને multiple inheritance કહેવામાં આવે છે. **C++ multiple inheritance** માં derived class માં એક કરતા વધુ base class હોય છે.
- multilevel inheritance થી multiple inheritance કેવી રીતે અલગ છે?
- **For example :**
    - **Multilevel inheritance :** child નું character એ Father દ્વારા Father નું character તેના પિતા (દાદા) દ્વારા

- **Multiple inheritance** : માતા અને પિતા તરફથી બાળક નું characters



- **C++ Multiple Inheritance Syntax :**

```
class A
{
  .....
};
class B
{
  .....
 } ;
class C : acess_specifier A,access_specifier A // derived class from A and B
{
  .....
} ;
```

Example :

```
#include<iostream>
using namespace std;
class A
{
        public:
        int x;
        void getx()
   {
            cout << "enter value of x: "; cin >> x;
   }
};
class B
{
        public:
        int y;
        void gety()
        {
            cout << "enter value of y: "; cin >> y;
        }
};
class C : public A, public B   //C is derived from class A and class B
{
        public:
        void sum()
        {
            cout << "Sum = " << x + y;
        }
};
```

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

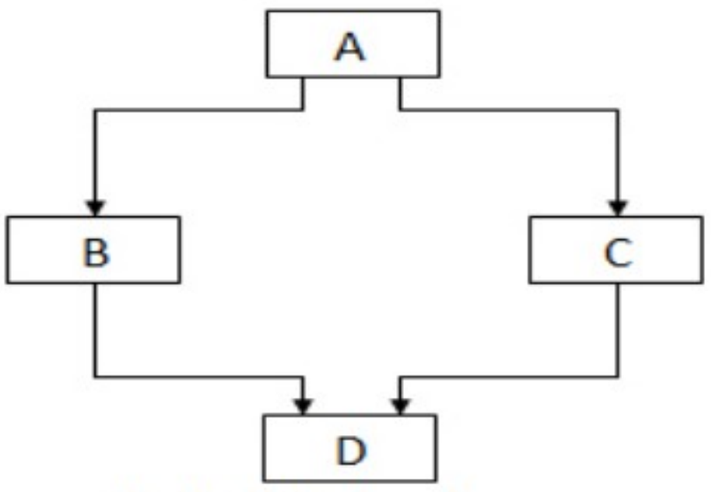
Output :

```
enter value of x:2
enter value of y:5
Sum =7
```

- ઉપર ના પ્રોગ્રામમાં, બે બેઝ class A અને B છે, જેમાંથી class C inherited કરવામાં આવ્યો છે. તેથી, derived class C એ A અને B ના તમામ public members ને inherits કરે છે અને તેમની visibility ને રાખે છે. અહીં, આપણે derived class C નો object obj1 બનાવ્યો છે.

6 Explain Hybrid Inheritance with example.

- જે inheritance માં class ના derivation માં કોઈપણ inheritance ના એકથી વધુ form આવેલા હોય છે તેને **hybrid inheritance** કહેવામાં આવે છે. Basically રીતે **C++** માં **hybrid inheritance** એ બે કે તેથી વધુ પ્રકારનાં inheritance નું combination છે. તેને multi path inheritance પણ કહી શકાય છે.
- એવી પરિસ્થિતિઓ હોઈ શકે છે કે જ્યારે કોઈ પ્રોગ્રામ ડિઝાઇન કરવા માટે આપણે બે અથવા વધુ પ્રકારનાં inheritance કરવાની જરૂર હોય તે સમયે આપણે એકથી વધુ inheritance નો ઉપયોગ કરવો પડશે. તેથી, આ પ્રકારના inheritance ને **Hybrid Inheritance** તરીકે ઓળખાય છે



Hybrid Inheritance

- ***Syntax :***

```
class A
{
    .....
};
class B : public A
{
    .....
} ;
class C
```

```
{
    .....
};
 class D : public B, public C
{
    .....
};
```

Example :

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

Output :

Sum=14

Program 2 :

```
#include<iostream>
class A
{
        public:
                void m_1()
                {
                cout<<"Hello, I am mehtod of Class A"<<endl;
                }
};
class B : public A
{
        public:
                void m_2()
                {
                cout<<"Hello, I am mehtod of Class B"<<endl;
                }
};
class D
{
        public:
                void m_4()
                {
                cout<<"Hello, I am mehtod of Class D"<<endl;
                }
};
class C: public B,public D
{
        public:
                void m_3()
                {
                cout<<"Hello, I am mehtod of Class C"<<endl;
                }
};
int main()
{
        clrscr();
        C obj;
        obj.m_1();
        obj.m_2();
        obj.m_3();
        obj.m_4();
        return 0;
}
```

Output :

```
Hello, I am mehtod of Class A
Hello, I am mehtod of Class B
Hello, I am mehtod of Class C
```

Hello, I am mehtod of Class D

7 Making private data inheritable.

- આપણે private data ને inherit કરી શકતા નથી.
- આપણે તેને public બનાવીને inherit કરી શકીએ છીએ, પરંતુ તેને public કર્યા પછી કોઈપણ ગમે ત્યાંથી access કરી શકે છે.
- C++ નું એક નવું access modifier એ **protected** છે.
- Private ડેટા ને protected રાખીને આપણે તેને inherit કરી શકીએ છીએ, પરંતુ આપણે બહાર access કરી શકતા નથી.
- private અને protected વચ્ચે ફક્ત એક જ તફાવત છે private એ inheritable નથી જ્યાં protected inheritable છે.

Specifiers	Within Same Class	In Derived Class	Outside the Class
Private	Yes	No	No
Protected	Yes	Yes	No
Public	Yes	Yes	Yes

Example :

```
#include <iostream>
using namespace std;
class A
{
        public:int a;
        protected:
        int b;
        public:
        int c;
        void init()
        {
                a=10;
                b=20;
                c=30;
        }
};
class B:public A
{
        public:
        void display()
        {
                cout<<a<<b<<c;
        }
};
int main()
{
        B bb;
```

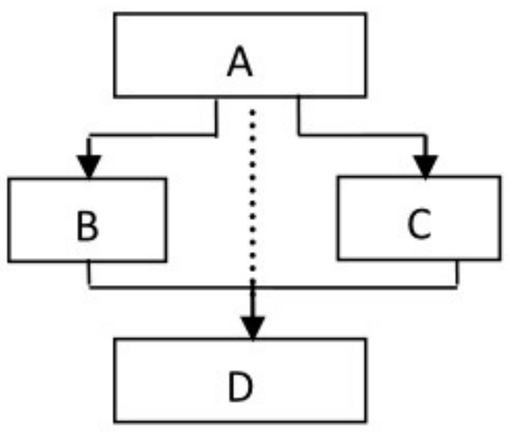
```
        bb.init();
        bb.display();
}
```

Output :

102030

8 Explain virtual base class with example.

- તેનો ઉપયોગ ડુપ્લિકેશન રોકવા માટે થાય છે.
- hybrid inheritance માં child class માં બે direct parents હોય છે જેને પોતાનો common base class હોય છે.
- તેથી, child class બે અલગ અલગ પાથ દ્વારા grandparent ને inherit કરે છે. તેને as indirect parent class પણ કહેવામાં આવે છે.
- grandparent ના બધા public અને protected member ને બે વાર child માં inherited કરવામાં આવ્યું છે.
- આપણે **virtual base class** બનાવીને આ ડુપ્લિકેશનને રોકી શકીએ છીએ.



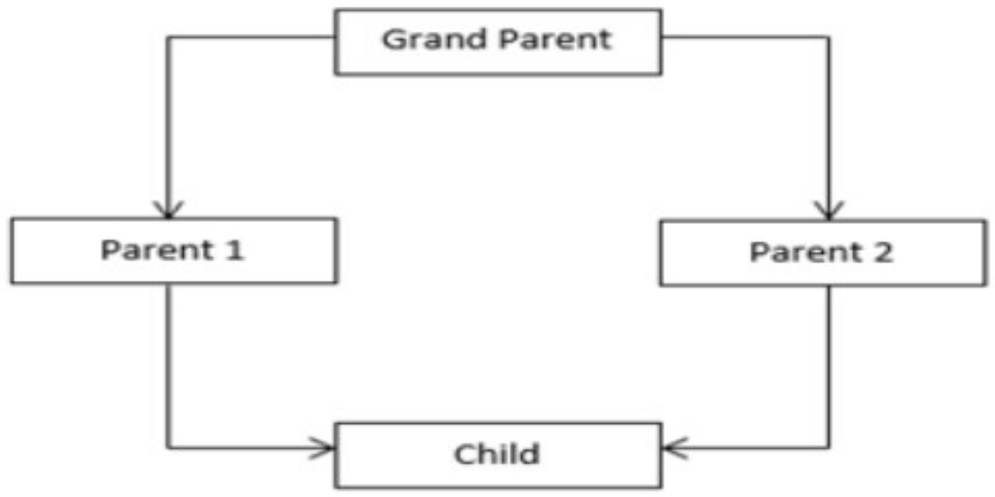


Figure: Multipath Inheritance

- આપણે figure પરથી જોઈ શકીએ છીએ કે class A ના data members/function બે વાર class D માં inherited થાય છે એક class B દ્વારા અને બીજો class C દ્વારા. જ્યારે class A ના કોઈપણ data / function member એ class D ના object દ્વારા એક્સેસ કરવામાં આવે છે ,ત્યારે ambiguity થાય છે કે કયા data/function member ને call કરવામાં આવશે? એક B દ્વારા inherited થાય છે અથવા બીજું C દ્વારા inherited થાય છે. આ કમ્પાઇલરને confuses કરે છે અને તે error બતાવે છે.

Virtual Base class નો ઉપયોગ કર્યા વગર નું Example :

```
#include <iostream>
using namespace std;
class A {
public:
    void show()
    {
            cout << "Hello form A \n";
    }
};

class B : public A {
};

class C : public A {
```

```
};

class D : public B, public C {
};

int main()
{
      D object;
      object.show();
    }
```

Output :Error

```
prog.cpp: In function 'int main()':
prog.cpp:29:9: error: request for member 'show' is ambiguous
  object.show();
         ^
prog.cpp:8:8: note: candidates are: void A::show()
   void show()
        ^
prog.cpp:8:8: note: void A::show()
```

How to resolve this issue?

આ ambiguity ને resolve કરવા માટે જ્યારે class A ને class B અને class C બંનેમાં inherited કરવામાં આવ્યું છે, તો કીવર્ડ virtual મૂકીને તેને virtual base class તરીકે declared કરવામાં આવે છે

Syntax :

```
class B : virtual public A
{
};
```

Virtual Base Class નો ઉપયોગ કરેલું **Example** :

```
#include <iostream>
using namespace std;
class A {
public:
        void show()
        {
                cout << "Hello from A \n";
        }
};
class B : virtual public A {
};
class C : virtual public A {
};
class D : public B, public C {
};
int main()
{
        D object;
```

```
        object.show();
}
```

Output :

Hello from A

9 Explain abstract class.

- C++ programming માં, કેટલીક વાર inheritance નો ઉપયોગ ફક્ત ડેટાના વધુ સારા વિઝ્યુલાઇઝેશન માટે થાય છે અને તમારે બેઝ ક્લાસનું કોઈ object બનાવવાની જરૂર નથી.
- ઉદાહરણ તરીકે: જો તમે તે circle અને square જેવા અલગ અલગ objects ના area ની ગણતરી કરવા માંગતા હો, તો તમે આ classes ને તેના shape થી inherit કરી શકો છો કારણ કે તે problem ને visualize કરવામાં મદદ કરે છે પરંતુ, તમારે shape ના કોઈ object બનાવવાની જરૂર પડતી નથી.
- આવા case માં, તમે abstract class તરીકે *shape* ને declare કરી શકો છો. જો તમે abstract class નો object બનાવવાની try કરશો, તો કમ્પાઇલર error બતાવે છે
- જો Expression=0 ને virtual function માં add કરવામાં આવે છે, તો તે function pure virtual function બની જાય છે.
- જો base class માં ઓછામાં ઓછું એક virtual function હોય, તો તે class એ abstract class તરીકે ઓળખાય છે.
- નીચેના પ્રોગ્રામમાં બતાવ્યા પ્રમાણે, pure virtual function એ virtual float area () = 0; class *Shape* ની અંદર define થયેલ છે, તેથી આ class એક abstract class છે અને તમે class *Shape* નો object બનાવી શકતા નથી.

Example :

```
#include<iostream>
using namespace std;
class shape
{
        protected:
                float l;
        public:
                void get_data()
                {
                        cin>>l;
                }
                virtual float area()=0;
};
class square : public shape
{
        public:
                float area()
                {
                        return l*l;
                }
};
```

```
class circle : public shape
{
        public:
                float area()
                {
                        return 3.14*l*l;
                }
};
int main()
{
        square s;
        circle c;
        cout<<"Enter the lenght to calculate area of square:"<<endl;
        s.get_data();
        cout<<"Area of a square is: "<<s.area();
        cout<<"\nEnter radius to calculate area of a circle:";
        c.get_data();
        cout<<"Area of Circle: "<<c.area();
        return 0;
}
```

Output :

Enter the lenght to calculate area of square: 5
Area of a square is:25
Enter radius to calculate area of a circle:2
Area of Circle:12.56

10 Explain Constructors in derived class with example.

- જ્યારે પણ class નો કોઈ object બનાવવામાં આવે ત્યારે Constructor automatically call થાય છે, પરંતુ inheritance માં માત્ર derived class માં જ object હોય છે.
- તેથી, જ્યારે પણ derived class નું object બને છે ત્યારે એ પહેલા base class ના constructor ને execute કરશે ત્યારબાદ derived class ના constructor ને execute કરશે.
- જો If base class ના constructor પાસે argument હોય તો આપણે નીચેની method દ્વારા derived class constructorપાસેથી આવી રીતે argument પાસ કરવી પડશે
-

Syntax :

```
class A
{
    A(int a)
    {
        Statement 1;
        Statement 2;
         ...
        Statement n;
    }
};
```

```
class B:public A
{
      B(int x,int y):A(x)
     {
         Statement 1;
         Statement 2;
           …
         Statement n;
      }
    };
```

- ઉપરના syntax માં આપણે derived class constructor થી base class constructor ની argument ને pass કરી છે.

Example :

```
#include <iostream>
using namespace std;
class A
{
        public:
        A(int a)
        {
                cout<<"\nValue of a="<<a;
        }
};
class B:public A
{
        public:
        B(int a,int b):A(a)
        {

                cout<<"\nValue of b="<<b;
        }
};
int main()
{
        B bb(10,20);
}
```

Output :

Value of a=10

Value of b=20