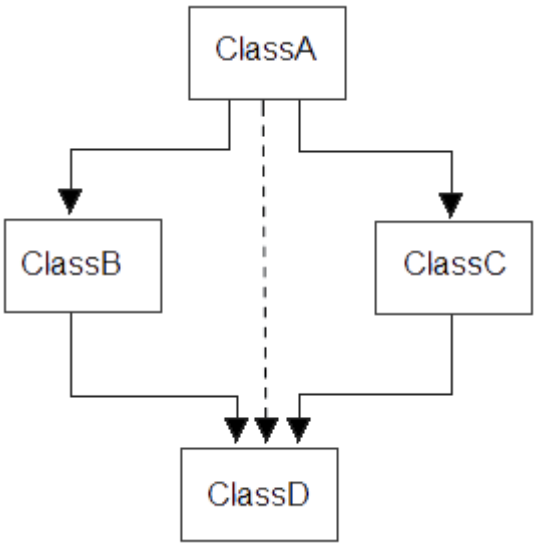
```
#include <iostream>
using namespace std;
class Base //base class
{
public:
void show()
{
 cout << “Base\n”;
}
};

class Derv1 : public Base
{
public:
void show()
{
 cout << “Derv1\n”;
 }
};

class Derv2 : public Base //derived class 2
{
public:
void show()
{
cout << “Derv2\n”;
}
};
int main()
{
Derv1 dv1;    //object of derived class 1
Derv2 dv2;    //object of derived class 2
Base* ptr;    //pointer to base class
ptr = &dv1;    //put address of dv1 in pointer
ptr->show();     //execute show()
ptr = &dv2;     //put address of dv2 in pointer
ptr->show() ;   //execute show()
return 0;
}
```

Virtual base class




```
#include<iostream>
using namespace std;
	class ClassA
	{
		public:
		int a;
	};

	class ClassB : public ClassA
	{
		public:
		int b;
	};
	class ClassC : public ClassA
	{
		public:
		int c;
	};
```

```
class ClassD : public ClassB, public ClassC
    {
        public:
        int d;
    };

     main()
    {
                    ClassD obj;

               //obj.a = 10;                //Statement 1, Error occur
                     //obj.a = 100;              //Statement 2, Error occur

                    obj.ClassB::a = 10;       //Statement 3
                    obj.ClassC::a = 100;      //Statement 4

                    obj.b = 20;
                    obj.c = 30;
                    obj.d = 40;

                    cout<< "\n A from ClassB  : "<< obj.ClassB::a;
                    cout<< "\n A from ClassC  : "<< obj.ClassC::a;

                    cout<< "\n B : "<< obj.b;
                    cout<< "\n C : "<< obj.c;
                    cout<< "\n D : "<< obj.d;

        }
```

Output :

```
A from ClassB  : 10
A from ClassC  : 100
B : 20
C : 30
D : 40
```

To avoid

```
class ClassB : virtual public ClassA
    {
        public:
        int b;
    };
    class ClassC : virtual public ClassA
    {
        public:
        int c;
    };
```