# Virtual Base Class in C++

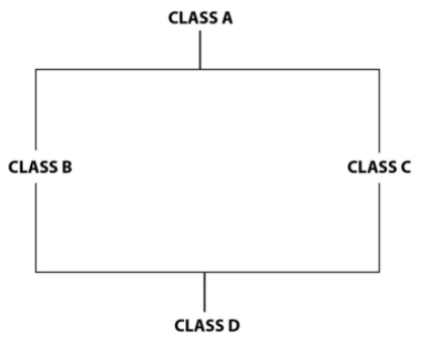
PRESENTED BY-

ABHILASHA SINGH

DEPARTMENT OF COMPUTER SCIENCE

# Introduction to Virtual Base Class

- Before getting into virtual base classes, let us revise the concepts of base class and inheritance.
- Base classes are the classes from which other classes are derived. The derived(child) classes have access to the variables and methods/ functions of a base(parent) class. The entire structure is known as inheritance hierarchy.



- Let us consider the above image, Class A is the parent class and Class B and C are the derived classes from Class A. Thus Class B and C has all the properties of class A.
- Next, Class D inherits Class B and also inherits Class C. With our knowledge Class D will get all the properties from Class B and C which also has Class A’s properties in them. What do you think will happen when we will try to access Class A’s properties through D? There will be an error because Class D will get Class A’s properties twice and compiler couldn’t decide what to output. You’ll see the term “ambiguous” when such a situation occurs.

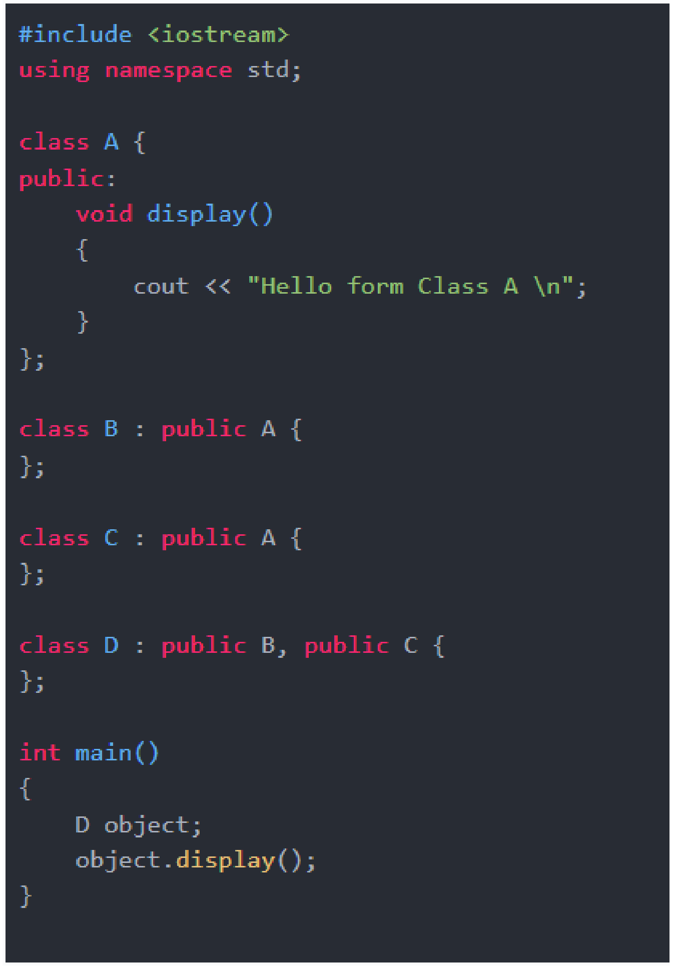
## WHAT IS VIRTUAL CLASS?

- Virtual Class is defined by writing a keyword “virtual” in the derived classes which allows only one copy of data to be copied to Class B and Class C(referring to the above example). It is a way of preventing multiple instances of a class appearing as a parent class in inheritance hierarchy when multiple inheritances are used.

## Need for Virtual Base Class in C++

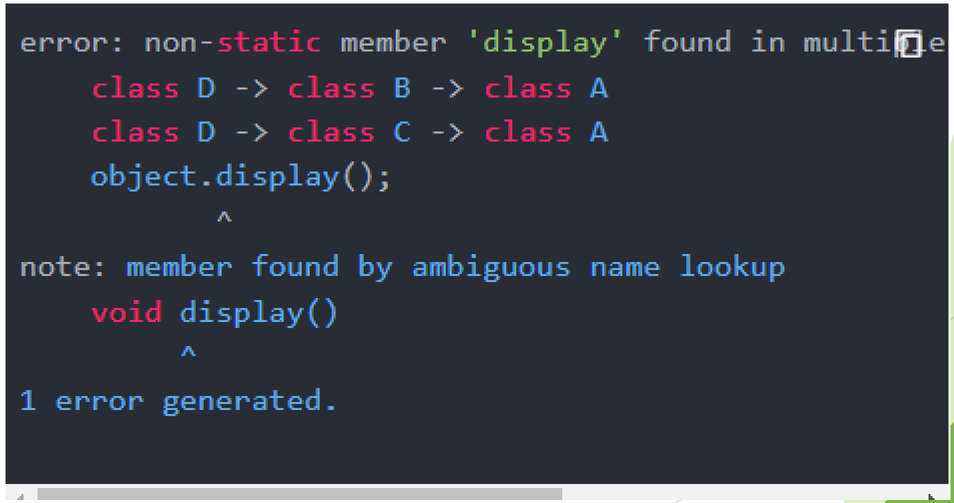
- In order to prevent the error and let the compiler work efficiently, we’ve to use virtual base class when multiple inheritance occurs. It saves space and avoid ambiguity.
- When a class is specified as a virtual base class, it prevents duplication of its data members. Only one copy of its data members is shared by all the base classes that use virtual base.
- If a virtual base class is not used, then all the derived classes will get duplicated data members. In this case the compiler cannot decide which one to execute. Let us look at an example without virtual base class in C++ and see how the output looks like:

# EXAMPLE



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

class A {
public:
    void display()
    {
        cout << "Hello form Class A \n";
    }
};

class B : public A {
};

class C : public A {
};

class D : public B, public C {
};

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

OUTPUT



```
error: non-static member 'display' found in multiple
    class D -> class B -> class A
    class D -> class C -> class A
    object.display();
           ^
note: member found by ambiguous name lookup
    void display()
         ^
1 error generated.
```

- In the above example, we create a class A and then two of its derived classes Class B and Class C. The Class A has has a method which prints out a statement. All the derived classes must have inherited data members from Class A.
- Next we declare Class D that inherits B and C. Since Class B and C are child classes of A and then D is the child class of B and C, Class D inherits data members of Class A from both B and C. Hence, duplication occurs and compiler doesn’t know what to execute and throws an error. But this situation is avoided if virtual base class is used.

## How to Declare Virtual Base Class in C++?

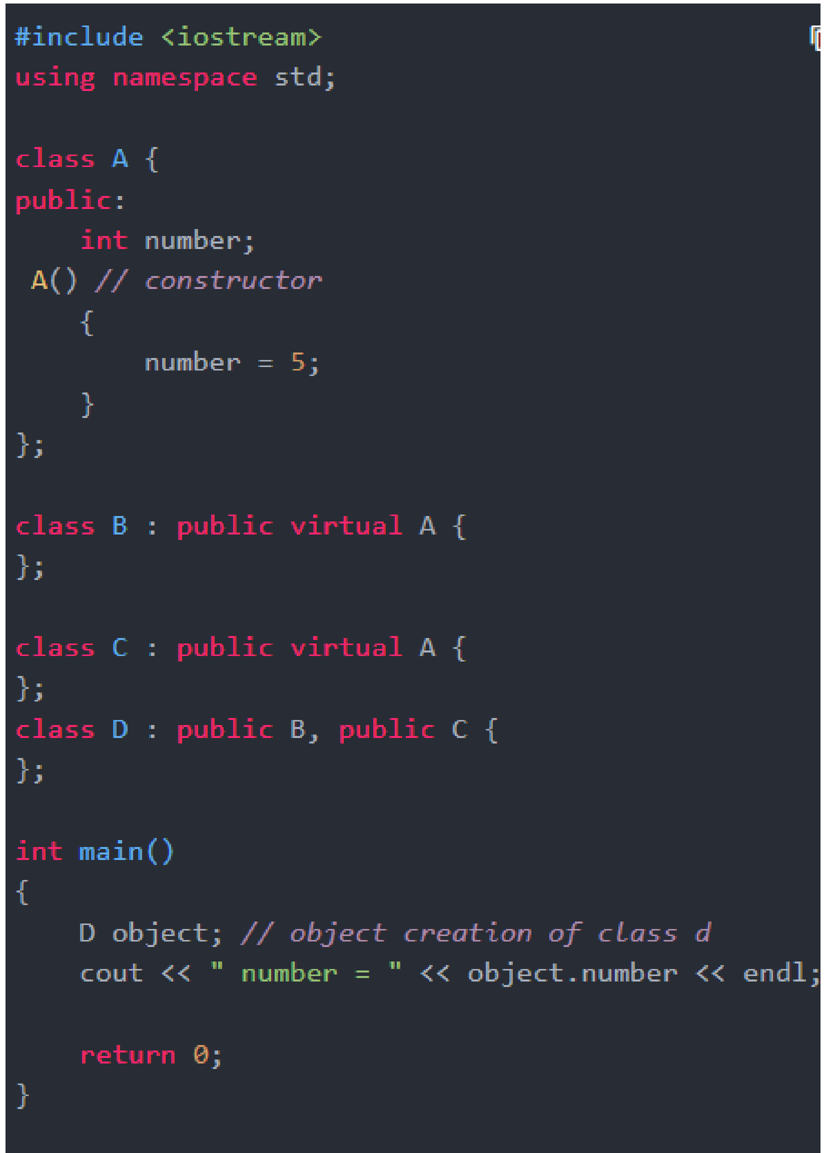
**Syntax**

- If Class A is considered as the base class and Class B and Class C are considered as the derived classes of A.

**Note:-** The word “virtual” can be written before or after the word “public”.

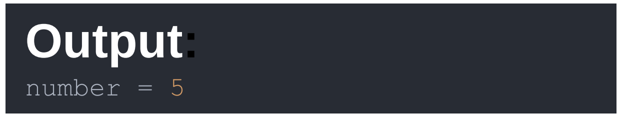
```
class B : virtual public A
{
/statement 1
};
class C : public virtual A
{
/statement 2
};
```

# EXAMPLE



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

class A {
public:
    int number;
 A() // constructor
    {
        number = 5;
    }
};

class B : public virtual A {
};

class C : public virtual A {
};
class D : public B, public C {
};

int main()
{
    D object; // object creation of class d
    cout << " number = " << object.number << endl;

    return 0;
}
```

- In this case, we are using virtual base class in C++ and so only one copy of data from Class A was inherited to Class D and hence the compiler will be able to print the output.
- When we mention the base class as virtual, we avoid the situation of duplication and let the derived classes get only one copy of data.



**Output**:

```
number = 5
```

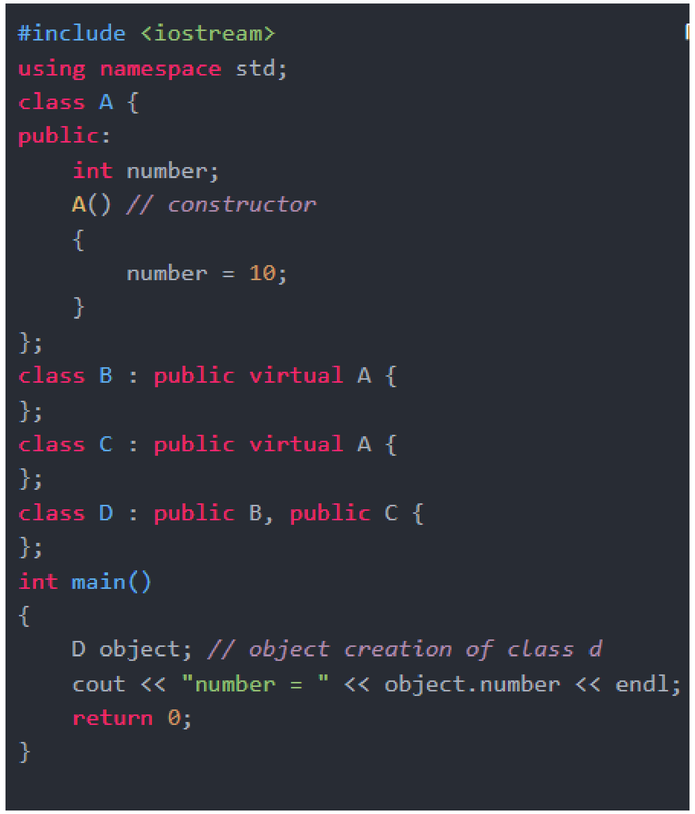
# A Pure Virtual Function

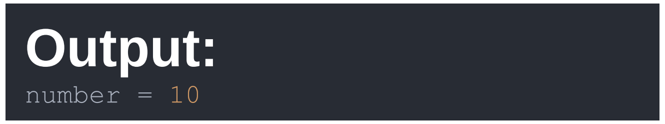
- A pure virtual function is a function that does nothing which means that you can declare a pure virtual function in the base class that does not have a description in the base class.
- Let's take an example of a class Animal(base class) that doesn't have implementation of moving but all the animals(derived classes) must know how to move considering that all animals can move.

**Syntax for Pure Virtual Function C++**

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

## EXAMPLE


```
#include <iostream>
using namespace std;
class A {
public:
    int number;
    A() // constructor
    {
        number = 10;
    }
};
class B : public virtual A {
};
class C : public virtual A {
};
class D : public B, public C {
};
int main()
{
    D object; // object creation of class d
    cout << "number = " << object.number << endl;
    return 0;
}
```


**Output:**

```
number = 10
```

# THANK YOU