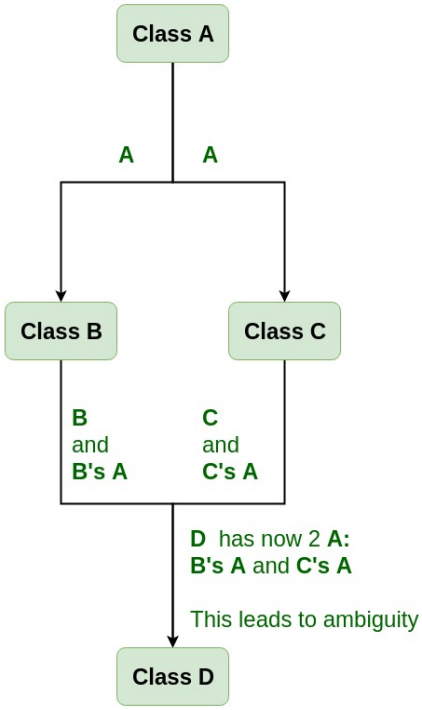
Unit 2:

- **Virtual base class in C++**

Virtual base classes in C++ are used to prevent multiple instances of a given class from appearing in an inheritance hierarchy when using multiple inheritances. Virtual Class is defined by writing a keyword “virtual” in the derived classes, allowing only one copy of data to be copied. Virtual Class is defined by writing a keyword “virtual” in the derived classes, allowing only one copy of data to be copied.

Example:



As we can see from the figure that data members/function of class **A** are inherited twice to class **D**. One through class **B** and second through class **C**. When any data / function member of class **A** is accessed by an object of class **D**, ambiguity arises as to which data/function member would be called? One inherited through **B** or the other inherited through **C**. This confuses compiler and it displays error.

Need for Virtual Base Class

To prevent the error and let the compiler work efficiently, we’ve to use a virtual base class when multiple inheritances occur. It saves space and avoids 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 the virtual base class.

If a virtual base class is not used, all the derived classes will get duplicated data members. In this case, the compiler cannot decide which one to execute.

Example: To show the need of Virtual Base Class in C++

```
#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();
}
```

Compile Errors:

```
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?

To resolve this ambiguity when class **A** is inherited in both class **B** and class **C**, it is declared as **virtual base class** by placing a keyword **virtual** as :

Syntax for Virtual Base Classes:
Syntax 1:

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

Syntax 2:

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

Example :

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

class A {
public:
        int a;
        A() // constructor
        {
                a = 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 << "a = " << object.a << endl;

        return 0;
}
```

Output

```
Hello from A
```