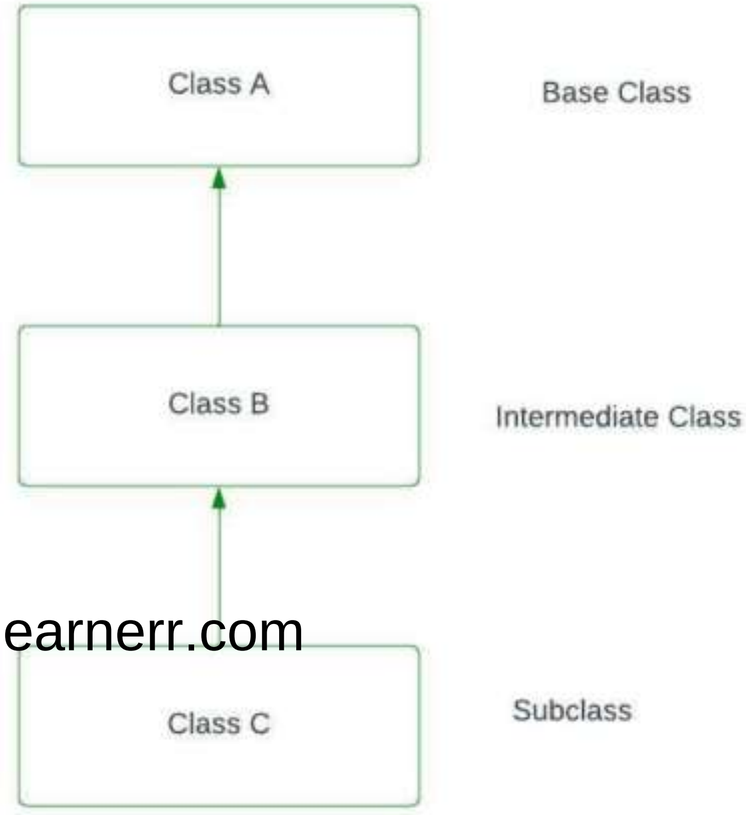
Inheritance

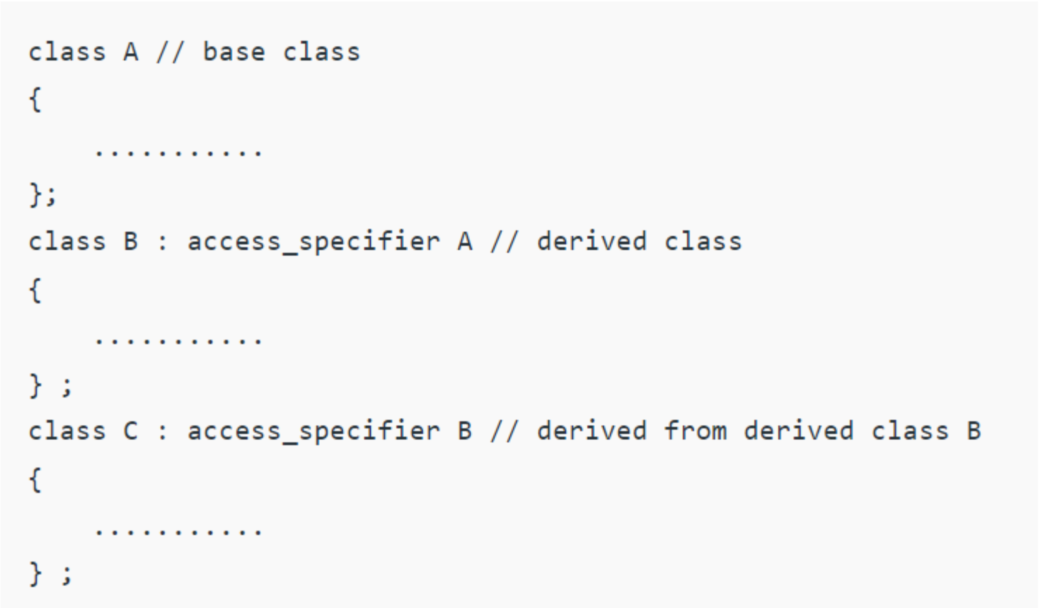
Multi-Level Inheritance

Multilevel Inheritance in C++ is the process of deriving a class from another derived class. When one class inherits another class it is further inherited by another class. It is known as multi-level inheritance.

For example, if we take Grandfather as a base class then Father is the derived class that has features of Grandfather and then Child is the also derived class that is derived from the sub-class Father which inherits all the features of Father.

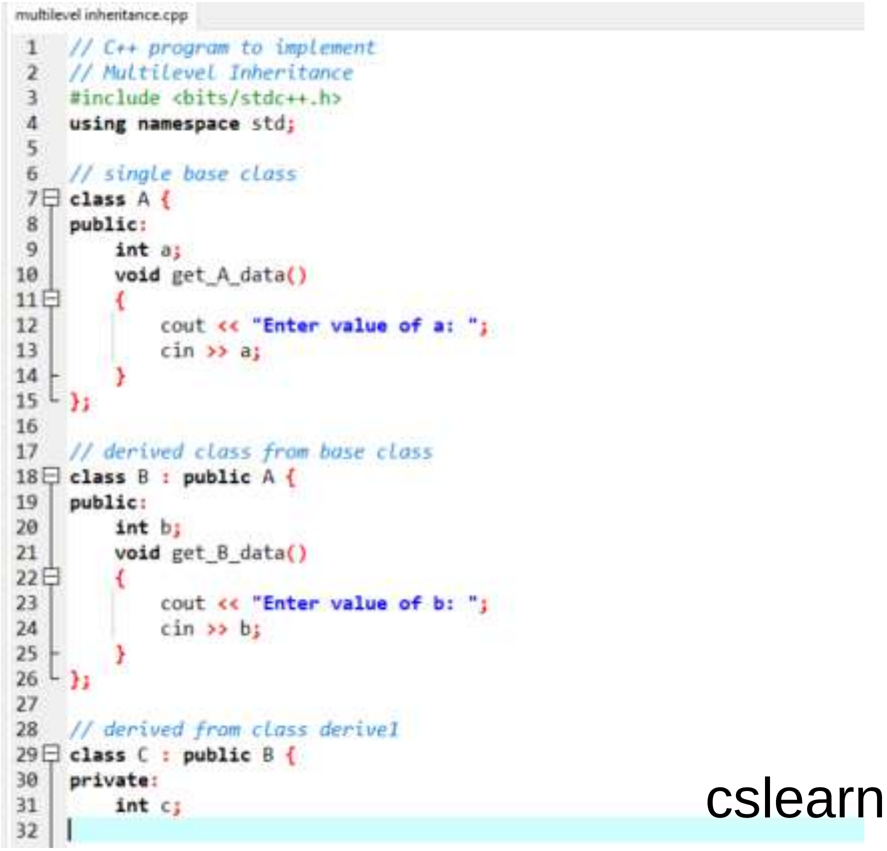
Multi-Level Inheritance



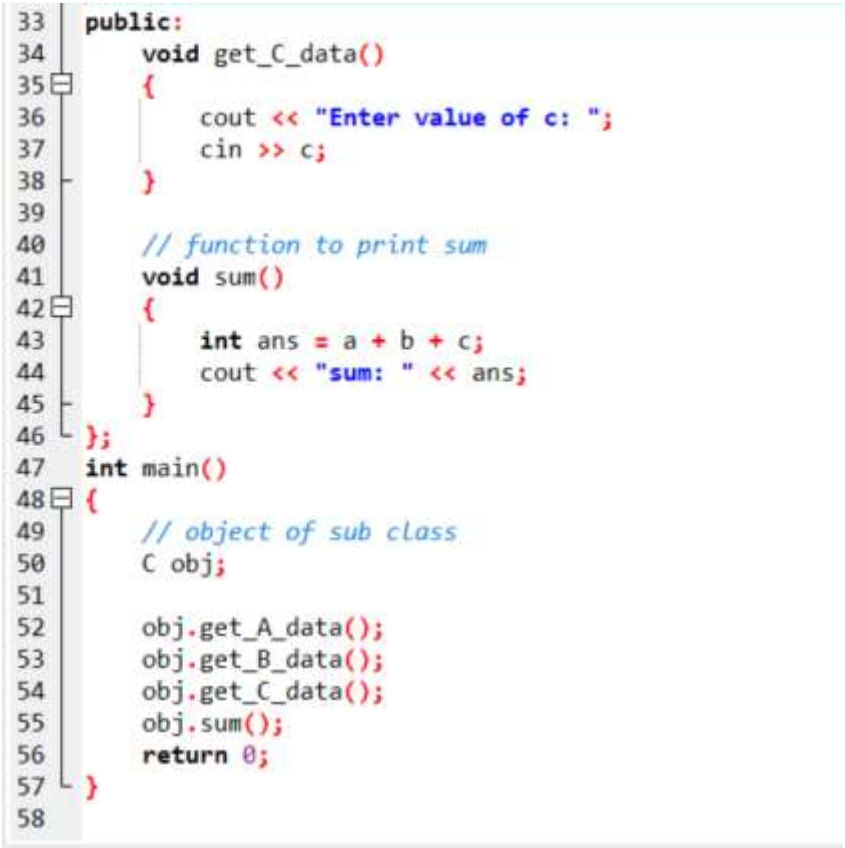


```
class A // base class
{
    ...........
};
class B : access_specifier A // derived class
{
    ...........
} ;
class C : access_specifier B // derived from derived class B
{
    ...........
} ;
```

Example : Multi-Level code

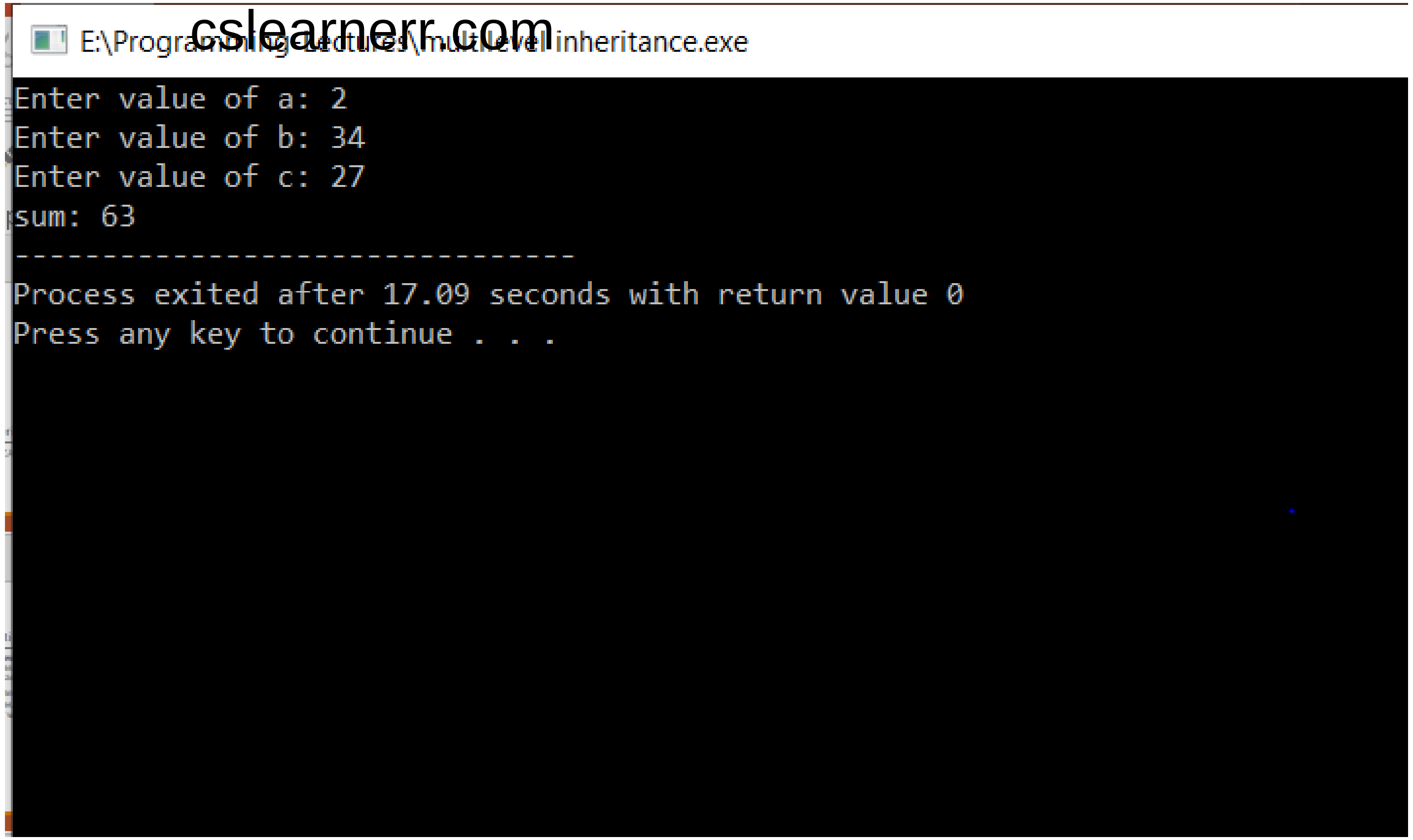


```
// C++ program to implement
// Multilevel Inheritance
#include <bits/stdc++.h>
using namespace std;

// single base class
class A {
public:
    int a;
    void get_A_data()
    {
        cout << "Enter value of a: ";
        cin >> a;
    }
};

// derived class from base class
class B : public A {
public:
    int b;
    void get_B_data()
    {
        cout << "Enter value of b: ";
        cin >> b;
    }
};

// derived from class derive1
class C : public B {
private:
    int c;

```



```
public:
    void get_C_data()
    {
        cout << "Enter value of c: ";
        cin >> c;
    }

    // function to print sum
    void sum()
    {
        int ans = a + b + c;
        cout << "sum: " << ans;
    }
};
int main()
{
    // object of sub class
    C obj;

    obj.get_A_data();
    obj.get_B_data();
    obj.get_C_data();
    obj.sum();
    return 0;
}

```

Output



E:\Programming Lectures\multilevel inheritance.exe

```
Enter value of a: 2
Enter value of b: 34
Enter value of c: 27
sum: 63
--------------------------------
Process exited after 17.09 seconds with return value 0
Press any key to continue . . .
```

Multiple Inheritance

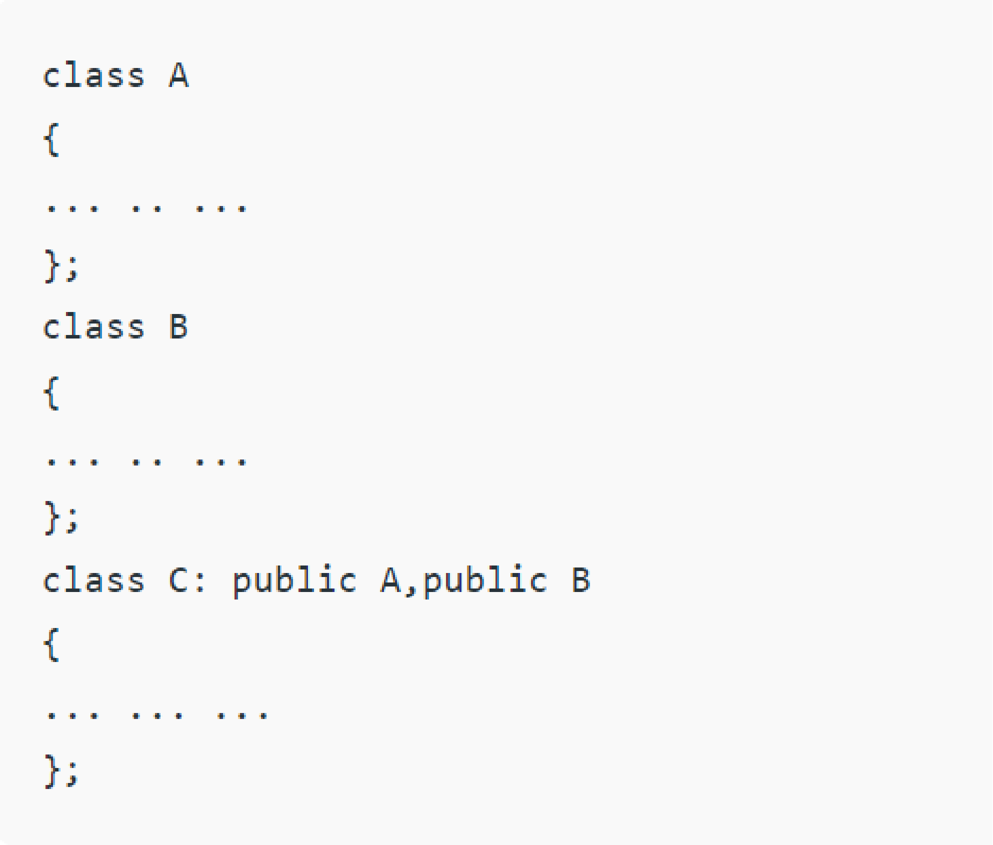
Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. The constructors of inherited classes are called in the same order in which they are inherited. For example, in the following program, B’s constructor is called before A’s constructor.

A class can be derived from more than one base class.

Eg:

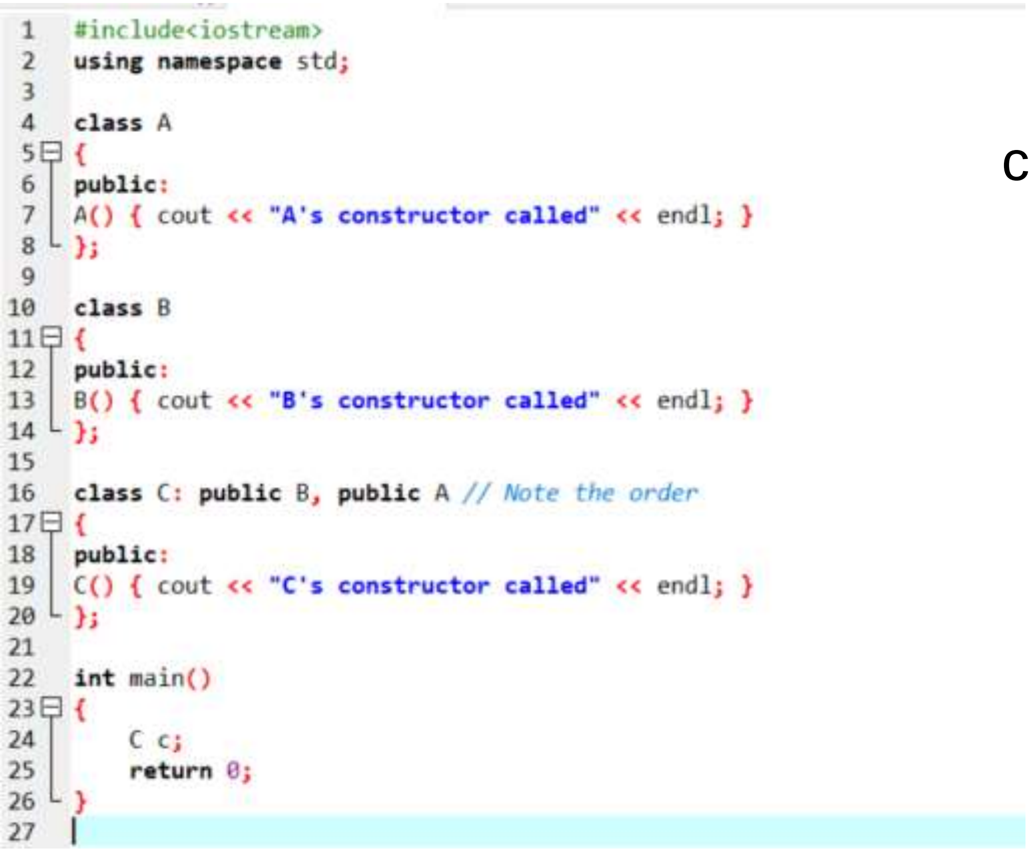
(i) A CHILD class is derived from FATHER and MOTHER class
(ii) A PETROL class is derived from LIQUID and FUEL class.

Syntax :

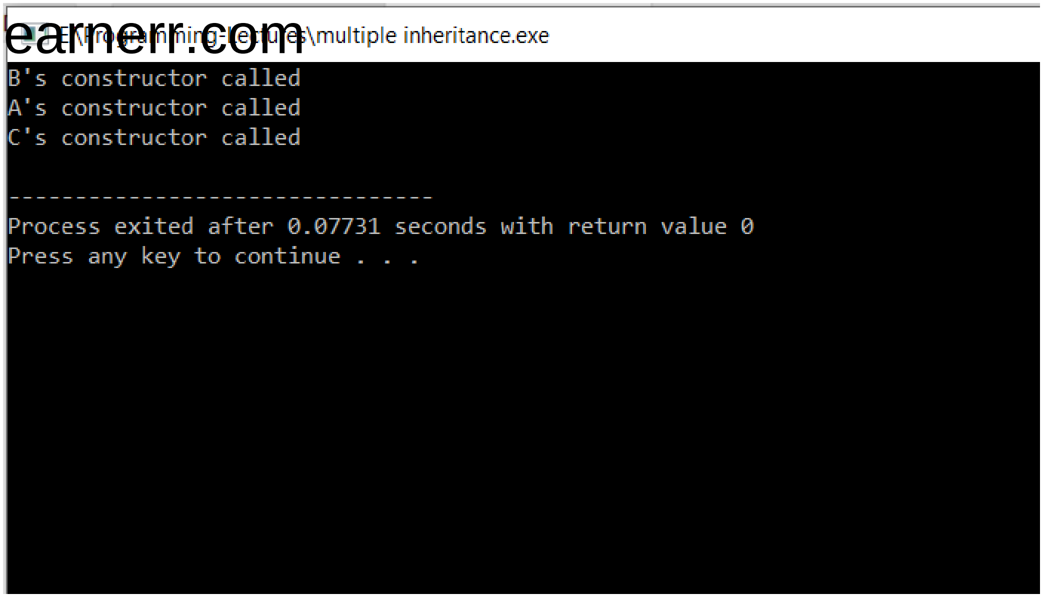


```
class A
{
... .. ...
};
class B
{
... .. ...
};
class C: public A,public B
{
... ... ...
};
```

Example:



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

class A
{
public:
A() { cout << "A's constructor called" << endl; }
};

class B
{
public:
B() { cout << "B's constructor called" << endl; }
};

class C: public B, public A // Note the order
{
public:
C() { cout << "C's constructor called" << endl; }
};

int main()
{
    C c;
    return 0;
}

```



```
B's constructor called
A's constructor called
C's constructor called

--------------------------------
Process exited after 0.07731 seconds with return value 0
Press any key to continue . . .
```

Hierarchical Inheritance

In Hierarchical inheritance, more than one sub-class inherits the property of a single base class. There is one base class and multiple derived classes.

Several other classes inherit the derived classes as well.

Hierarchical structures thus form a tree-like structure.

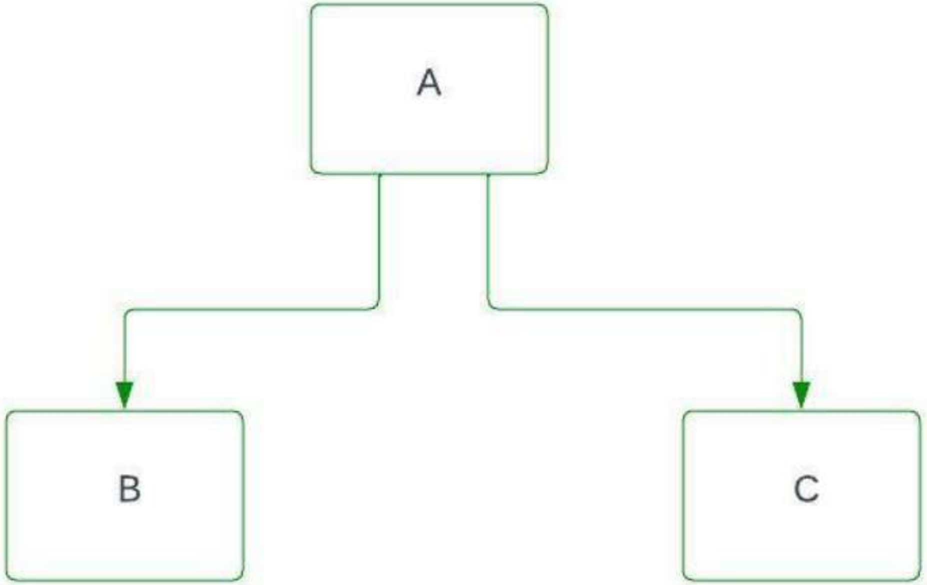
It is similar to that, mango and apple both are fruits; both inherit the property of fruit.

Fruit will be the Base class, and mango and apple are sub-classes.

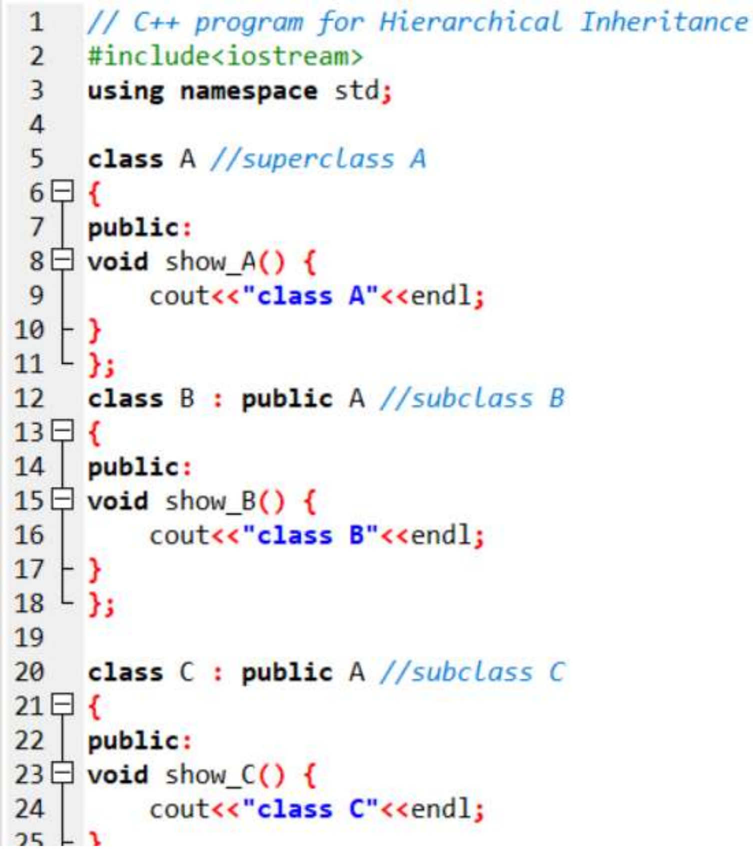
Hierarchical Inheritance

The below diagram shows, Class A is a Base class, B is a subclass inherited from class A, and C is a subclass it also inherits from class A.

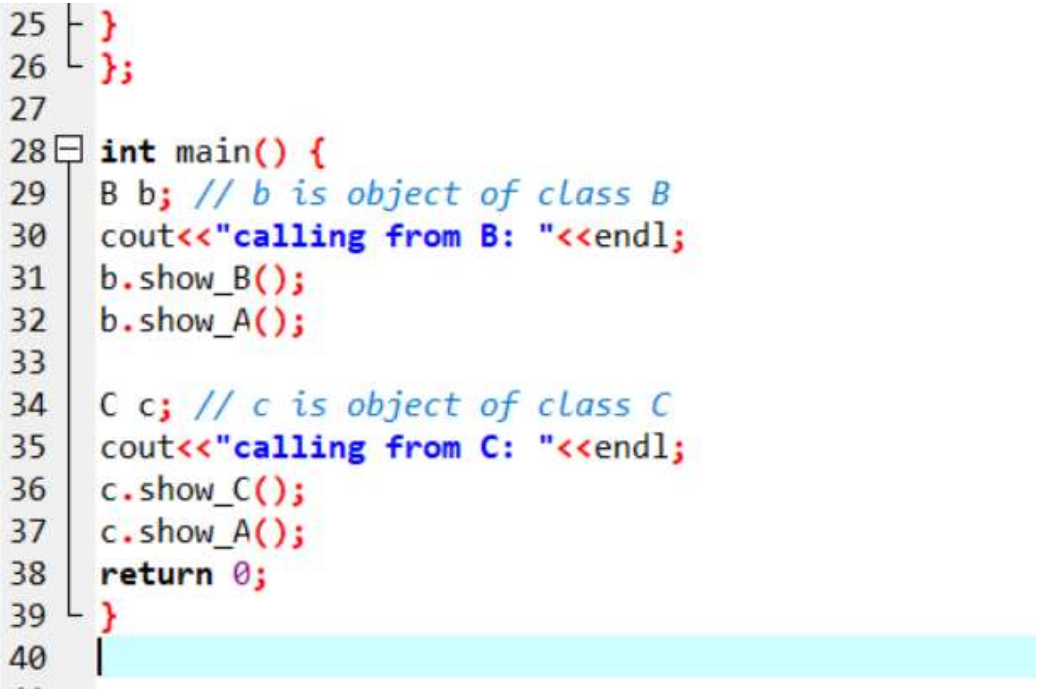
Similarly, if another subclass inherits property from B class and so on then there will be a hierarchy, and a tree-like structure is formed.



Example code

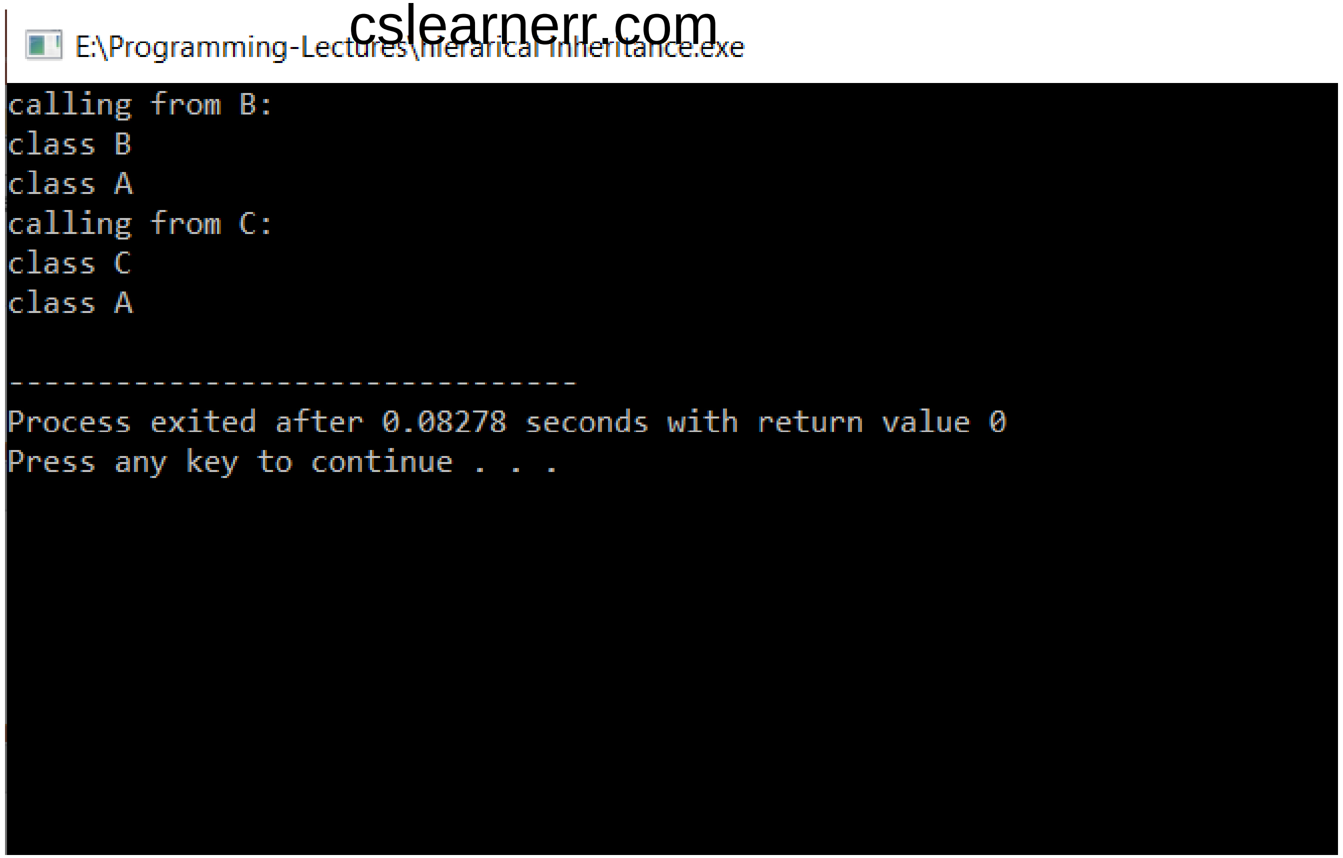


```
// C++ program for Hierarchical Inheritance
#include<iostream>
using namespace std;

class A //superclass A
{
public:
void show_A() {
     cout<<"class A"<<endl;
}
};
class B : public A //subclass B
{
public:
void show_B() {
     cout<<"class B"<<endl;
}
};

class C : public A //subclass C
{
public:
void show_C() {
     cout<<"class C"<<endl;
}
```



```
}
};

int main() {
B b; // b is object of class B
cout<<"calling from B: "<<endl;
b.show_B();
b.show_A();

C c; // c is object of class C
cout<<"calling from C: "<<endl;
c.show_C();
c.show_A();
return 0;
}
```

Example



```
calling from B:
class B
class A
calling from C:
class C
class A

--------------------------------
Process exited after 0.08278 seconds with return value 0
Press any key to continue . . .
```

Hybrid Inheritance

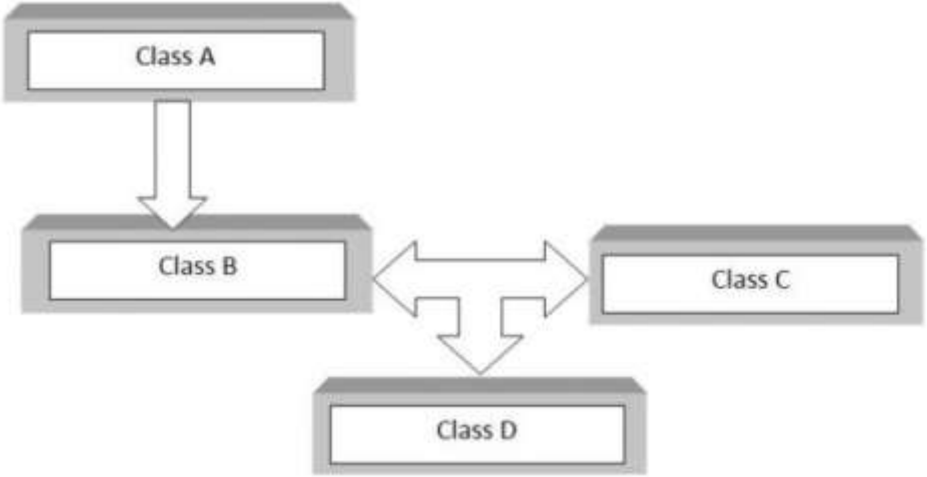
The inheritance in which the derivation of a class involves more than one form of any inheritance is called **hybrid inheritance**.

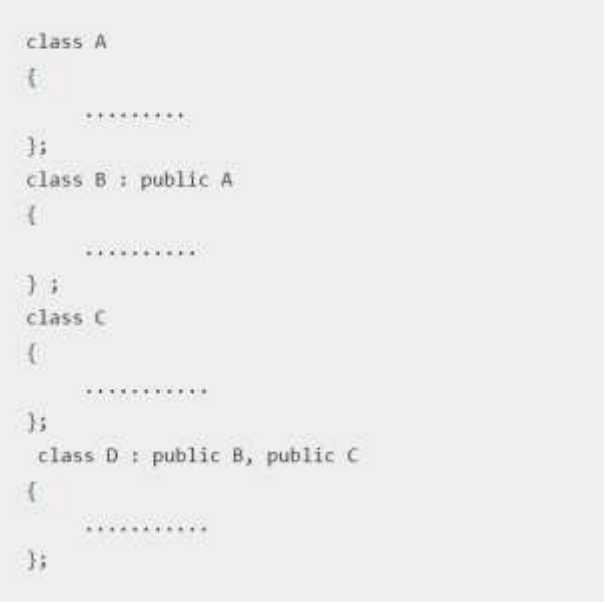
Basically **C++ hybrid inheritance** is combination of two or more types of inheritance.

It can also be called multi path inheritance.

The hybrid is combination of single inheritance and multiple inheritance. Hybrid inheritance is used in a situation where we need to apply more than one inheritance in a program.

Hybrid Inheritance




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

As shown in block diagram class B is derived from class A which is single inheritance and then Class D is inherited from B and class C which is multiple inheritance. So single inheritance and multiple inheritance jointly results in hybrid inheritance.