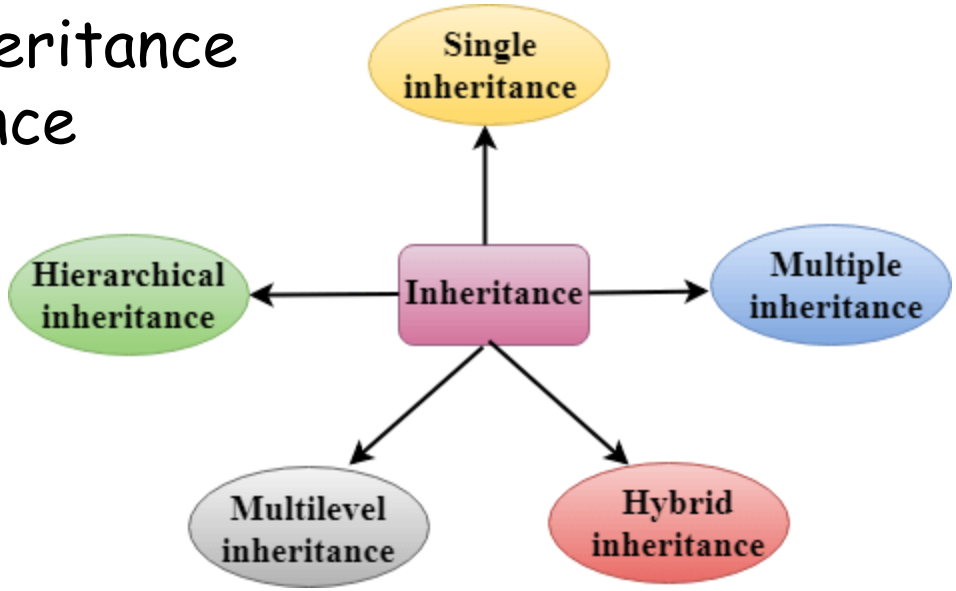
Types of Inheritance

Md. Sharif Hossen
Assistant Professor, Dept. of ICT, Comilla University,

Types of Inheritance

C++ supports five types of inheritance:

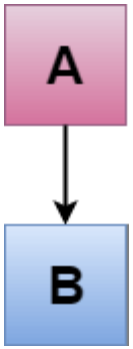
1. Single inheritance
2. Multilevel inheritance
3. Multiple inheritance
4. Hierarchical inheritance
5. Hybrid inheritance



Md. Sharif Hossen
Assistant Professor, Dept. of ICT, Comilla University,

Single Inheritance

Single inheritance is defined as the inheritance in which a derived class is inherited from the only one base class.



Where 'A' is the base class, and 'B' is the derived class.

Multilevel Inheritance

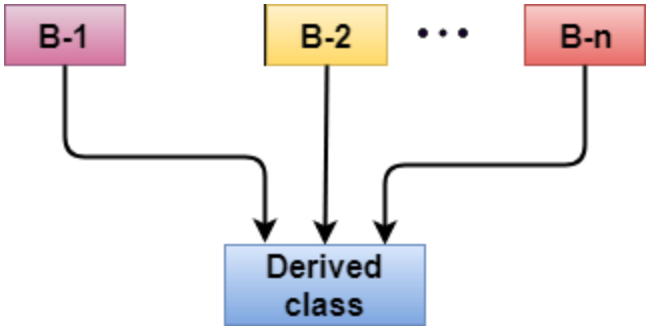
Multilevel inheritance is a process of deriving a class from another derived class.

When one class inherits another class which is further inherited by another class, it is known as multi level inheritance in C++. Inheritance is transitive so the last derived class acquires all the members of all its base classes.

Md. Sharif Hossen
Assistant Professor, Dept. of ICT, Comilla University,

Multiple Inheritance

Multiple inheritance is the process of deriving a new class that inherits the attributes from two or more classes.



Syntax of the Derived class:

```
class D : visibility B-1, visibility B-2, ?
{
    // Body of the class;
}
```

Ambiguity Resolution in Inheritance

Ambiguity can be occurred in using the multiple inheritance when a function with the same name occurs in more than one base class.

```
#include <iostream>
using namespace std;
class A
{
    public:
    void display()
    {
        cout << "Class A" << endl;
    }
};
class B
{
    public:
    void display()
    {
        std::cout << "Class B" << std::endl;
    }
};
```

```
class C : public A, public B
{
    public:
    void view()
    {
        display();
    }
};
int main()
{
    C c;
    //c.display();
    c.view();
    return 0;
}
```

```
 In member function 'void C::view()':
24:13: error: reference to 'display' is ambiguous
14:14: note: candidates are: void B::display()
6:14: note:                  void A::display()
```

Md. Sharif Hossen
Assistant Professor, Dept. of ICT, Comilla Universi

Ambiguity Resolution in Inheritance

The above issue can be resolved by using the class **resolution operator** with the function. In the above example, the derived class code can be rewritten as:

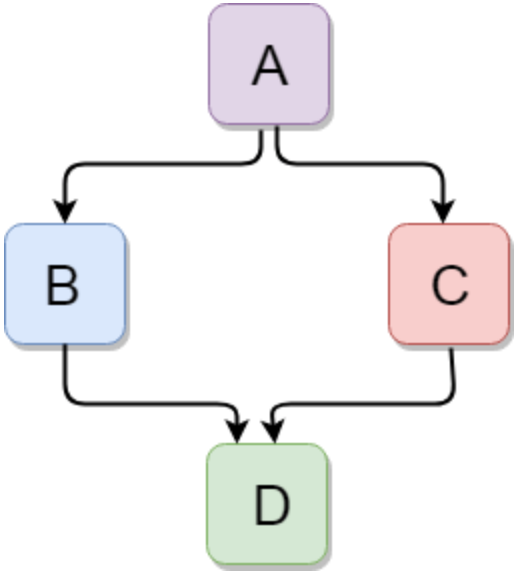
```cpp
#include <iostream>
using namespace std;
class A
{
    public:
    void display()
    {
        cout << "Class A" << std::endl;
    }
};
class B
{
    public:
    void display()
    {
        std::cout << "Class B" << std::endl;
    }
};
```

```cpp
class C : public A, public B
{
    public:
    void view()
    {
        A :: display();       // Calling the
            display() function of class A.
        B :: display();       // Calling the
            display() function of class B.

    }
};

int main()
{
    C c;
    //c.display();
    //c.A::display();
    c.view();
    return 0;
}
```

Md. Sharif Hossen
Assistant Professor, Dept. of ICT, Comilla University,

Hybrid Inheritance

Hybrid inheritance is a combination of more than one type of inheritance



Hybrid Inheritance

```
#include <iostream>
using namespace std;
class A
{
    protected:
    int a;
    public:
    void get_a(){
       std::cout << "Enter the value of 'a' : ";
       cin>>a;
    }
};

class B : public A
{
    protected:
    int b;
    public:
    void get_b()
    {
        std::cout << "Enter the value of 'b' : ";
       cin>>b;
    }
};
```

Hybrid Inheritance

```
class C
{
    protected:
    int c;
    public:
    void get_c()
    {
        std::cout << "Enter the value of c is : " ;
        cin>>c;
    }
};

class D : public B, public C
{
    protected:
    int d;
    public:
    void mul()
    {
        get_a();
        cout << a <<endl;
        get_b();
        cout << b <<endl;
        get_c();
        cout << c << endl;
        std::cout << "Multiplication of a,b,c is : " <<a*b*c<< std::endl;
    }
};
```

```
int main()
{
    D d;
    d.mul();
    return 0;
}
```

INPUT

```
100 200 300
```

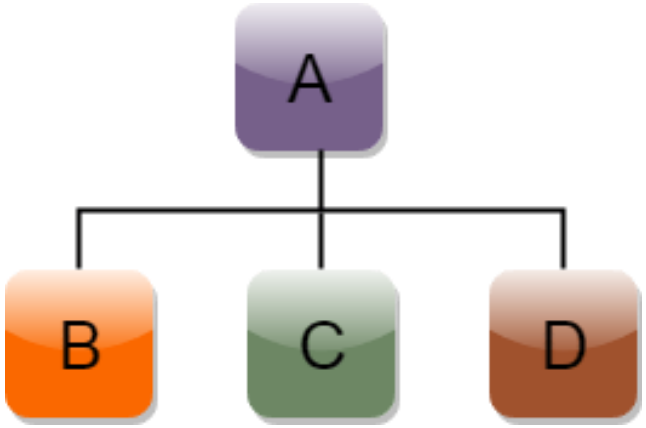
OUTPUT

```
Enter the value of 'a' : 100
Enter the value of 'b' : 200
Enter the value of c is : 300
Multiplication of a,b,c is : 6000000
```

Md. Sharif Hossen
Assistant Professor, Dept. of ICT, Comilla University,

Hierarchical Inheritance

Hierarchical inheritance is defined as the process of deriving more than one class from a base class.



Syntax:

```
class A
{
    // body of the class A.
}
class B : public A
{
    // body of class B.
}
class C : public A
{
    // body of class C.
}
class D : public A
{
    // body of class D.
}
```

Md. Sharif Hossen
Assistant Professor, Dept. of ICT, Comilla University,

Hierarchical Inheritance

```
#include <iostream>
using namespace std;
class Shape
{
    public:
    int a;
    int b;
    void get_data(int n,int m)
    {
        a= n;
        b = m;
    }
};
class Rectangle : public Shape
{
    public:
    int rect_area()
    {
        int result = a*b;
        return result;
    }
};
```

```
class Triangle : public Shape
{
    public:
    int triangle_area()
    {
        float result = 0.5*a*b;
        return result;
    }
};
```

Hierarchical Inheritance

```
int main()
{
    Rectangle r;
    Triangle t;
    int length,breadth,base,height;
    std::cout << "Enter the length and breadth of a rectangle: ";
    cin>>length>>breadth;
    cout << length << "   " << breadth << endl;

    r.get_data(length,breadth);
    int m = r.rect_area();
    std::cout << "Area of the rectangle is : " <<m<< std::endl;
    std::cout << "Enter the base and height of the triangle: ";
    cin>>base>>height;
    cout << base <<"   "<< height << endl;

    t.get_data(base,height);
    float n = t.triangle_area();
    std::cout <<"Area of the triangle is : "  << n<<std::endl;
    return 0;
}
```

INPUT

```
45 50
3 6
```

OUTPUT

```
Enter the length and breadth of a rectangle: 45   50
Area of the rectangle is : 2250
Enter the base and height of the triangle: 3   6
Area of the triangle is : 9
```