BAKI ALİ NEFT MƏKTƏBİ
BAKU HIGHER OIL SCHOOL

Programming and Computer Applications-2

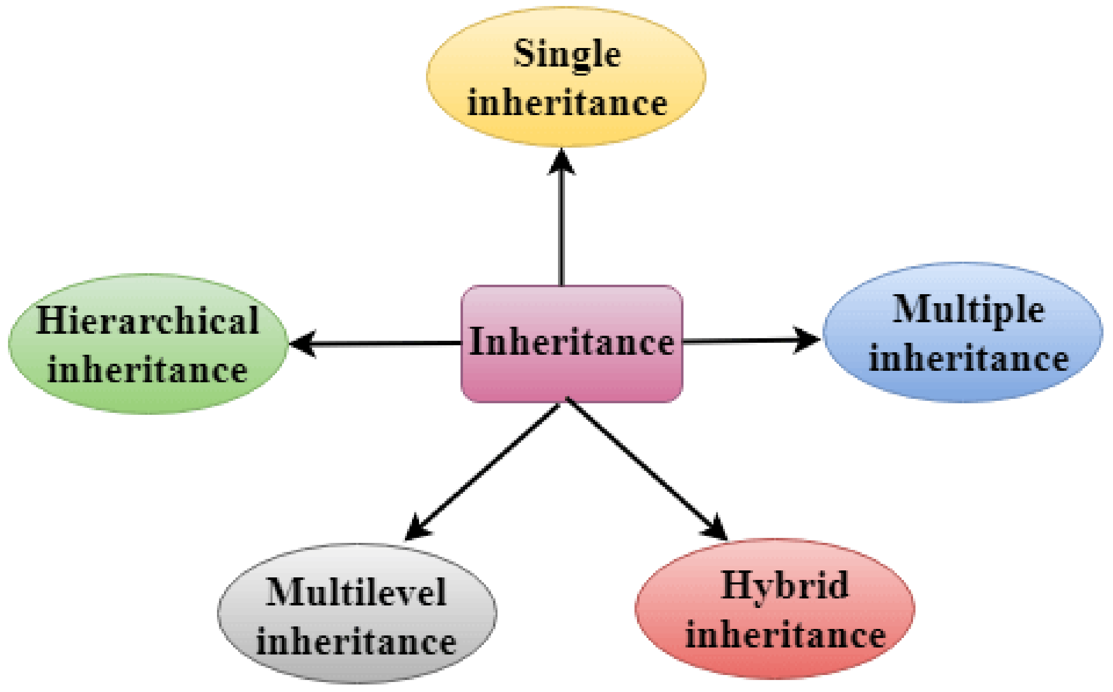
Different Types of Inheritance

Instructor : PhD, Associate Professor Leyla Muradkhanli

Types of Inheritance

C++ supports five types of inheritance:

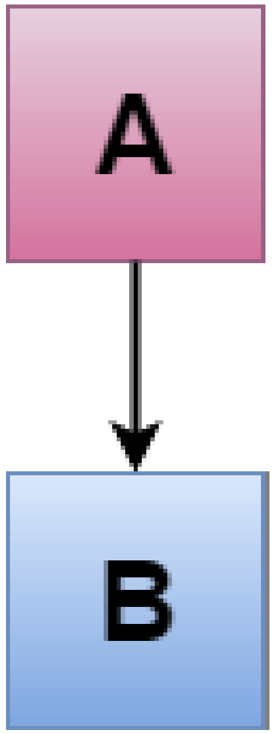
- Single inheritance
- Multiple inheritance
- Hierarchical inheritance
- Multilevel inheritance
- Hybrid inheritance



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.

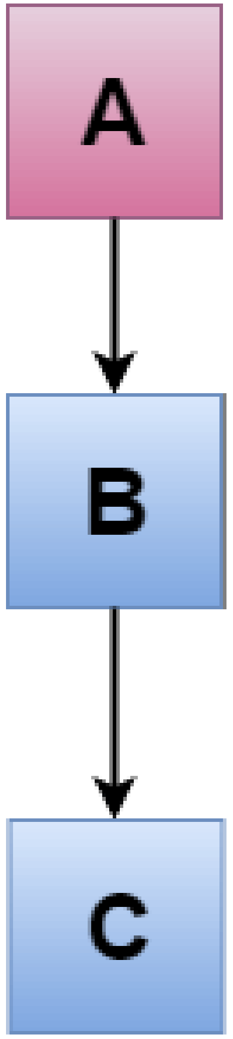


Single Inheritance

```
//Base Class
class A
{
... .. ...
};
//Derived Class
class B: public A
{
... .. ...
};
```

Multilevel Inheritance

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



Multilevel Inheritance

In C++ programming, not only you can derive a class from the base class but you can also derive a class from the derived class. This form of inheritance is known as multilevel inheritance.

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

Here, class B is derived from the base class A and the class C is derived from the derived class B.

```cpp
#include <iostream>
using namespace std;
class A
{
    public:
      void display()
      {
          cout<<"Base class content.";
      }
};
class B : public A
{
};
class C : public B
{
};
int main()
{
    C obj;
    obj.display();
return 0;
}
```

```
Output :
Base class content.
```

Multilevel Inheritance

In this program, class C is derived from class B (which is derived from base class A).

The obj object of class C is defined in the main() function.

When the display() function is called, display() in class A is executed. It's because there is no display() function in class C and class B.

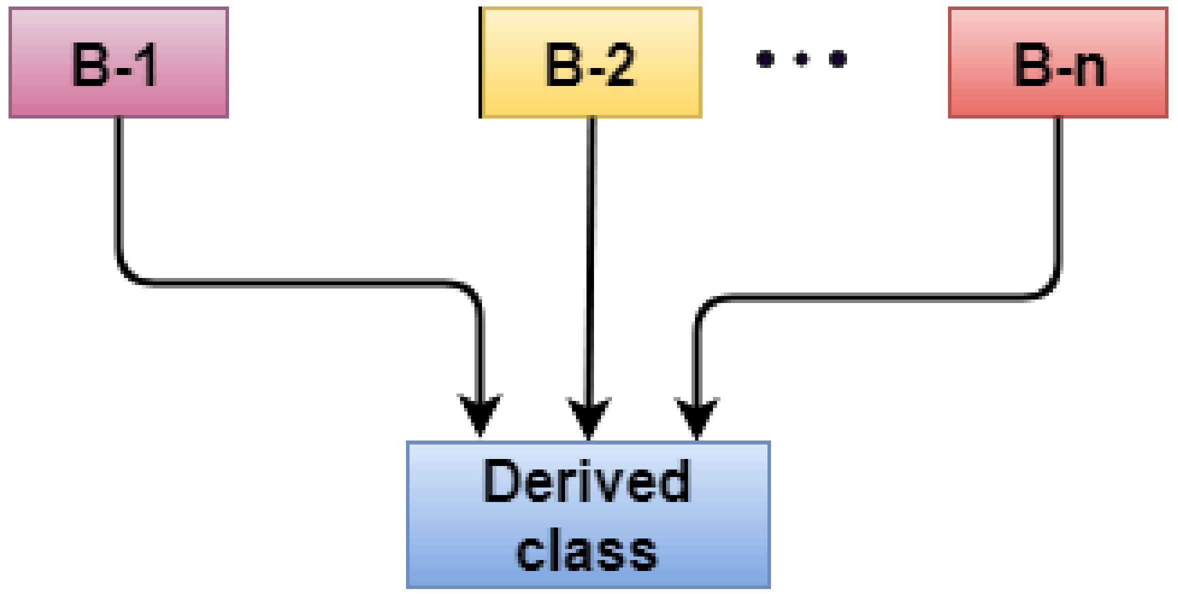
The compiler first looks for the display() function in class C. Since the function doesn't exist there, it looks for the function in class B (as C is derived from B).

The function also doesn't exist in class B, so the compiler looks for it in class A (as B is derived from A).

If display() function exists in C, the compiler overrides display() of class A (because of member function overriding).

Multiple Inheritances

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



Multiple Inheritances

A class may inherit from more than one class by simply specifying more base classes, separated by commas, in the list of a class's base classes (i.e., after the colon).

Ambiguity in Multiple Inheritances

The most obvious problem with multiple inheritance occurs during function overriding.

Suppose, two base classes have a same function which is not overridden in derived class.

If you try to call the function using the object of the derived class, compiler shows error. It's because compiler doesn't know which function to call. For example,

```
class base1
{
  public:
     void someFunction( )
     { .... ... .... }
};
class base2
{
    void someFunction( )
     { .... ... .... }
};

class derived : public base1, public base2
{
};
int main()
{
    derived obj;
    obj.someFunction() // Error!
}
```

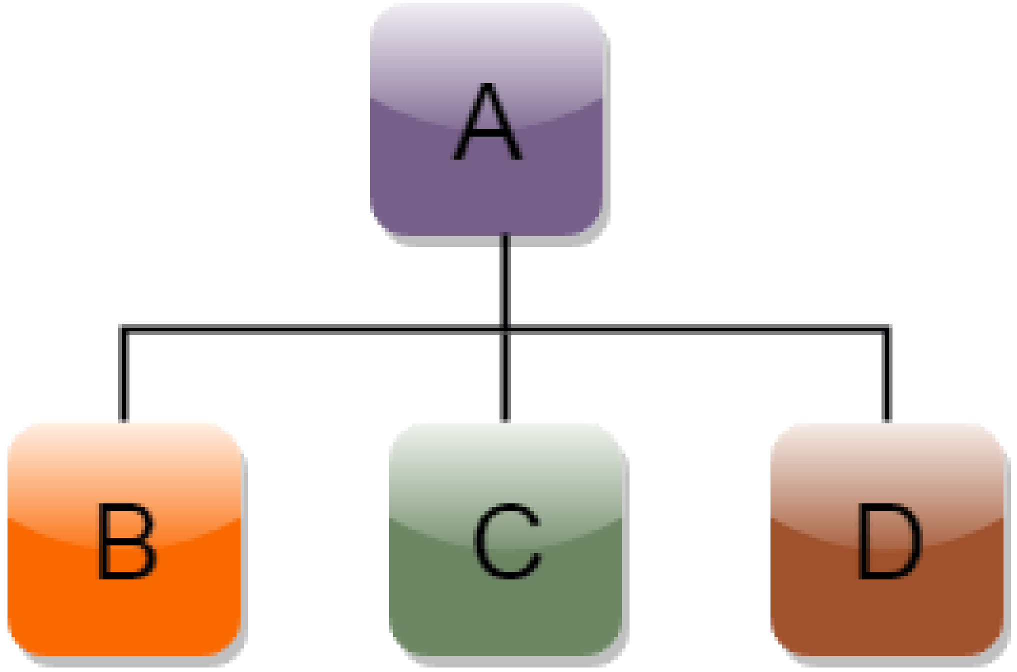
Ambiguity in Multiple Inheritances

This problem can be solved using scope resolution function to specify which function to class either *base1* or *base2*

```
int main()
{
   obj.base1::someFunction( );  // Function of base1 class is called
   obj.base2::someFunction();   // Function of base2 class is called.
}
```

Hierarchical Inheritance

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



Hierarchical Inheritance

If more than one class is inherited from the base class, it's known as hierarchical inheritance. In hierarchical inheritance, all features that are common in child classes are included in the base class.

For example: Physics, Chemistry, Biology are derived from Science class.

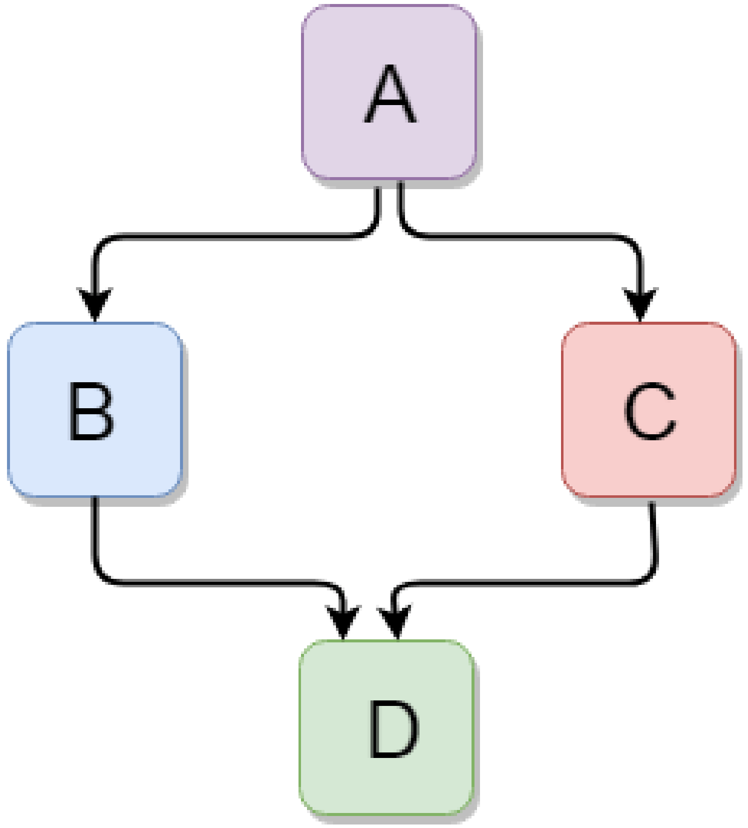
Syntax of Hierarchical Inheritance

```
 //Base Class
  class A
  {
 // body of the class A
}
  //Derived Class
  class B : public A
  {
// body of the class B
}
```

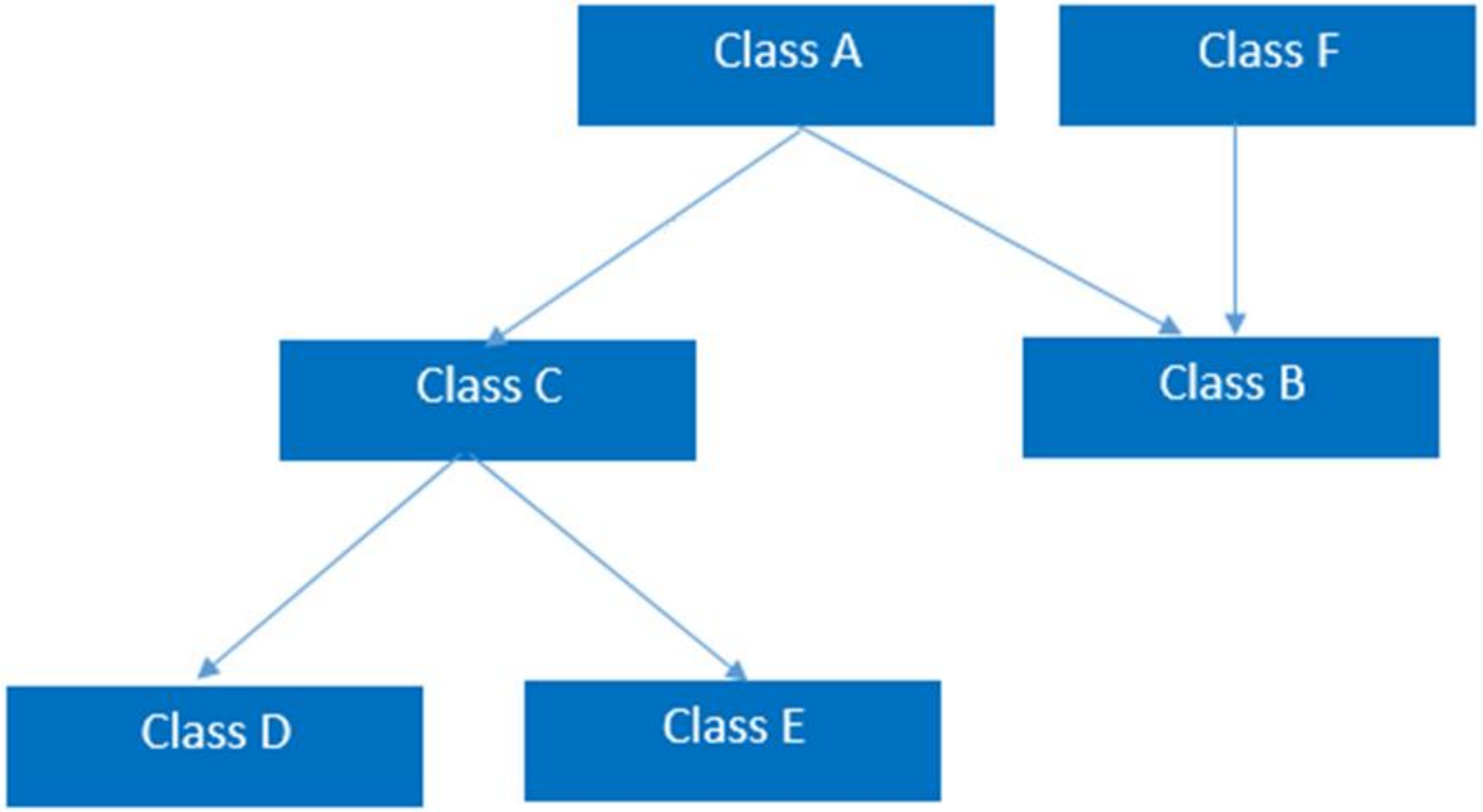
```
//Derived Class
  class C : public A
  {
// body of the class C
}
//Derived Class
  class D : public A
  {
// body of the class D
}
```

Hybrid Inheritance

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



Hybrid Inheritance



Hybrid Inheritance – (a combination of Hierarchical and multiple)

```
 //Base Class
  class  A
  {
}
  //Base Class
  class F
  {
}
  //Derived Class
    class  B : public A, public F
  {
}
  //Derived Class
  class  C : public A
  {
}
  //Derived Class
  class D : public C
  {
}
  //Derived Class
  class  E : public C
  {
}
```

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

```
 class C
{
    protected:
    int c;
    public:
    void getc()
    {
        cout << "Enter the value of c is : " <<endl;
        cin>>c;
    }
};
class D : public B, public C
{
    protected:
    int d;
    public:
    void mult()
    {
         geta();
         getb();
         getc();
         cout << "Multiplication of a,b,c is : " <<a*b*c<<endl;
    }
};
```

```
 int main()
{
    D obj;
    obj.mult();
return 0;
}
```