**Inheritance**

In C++, inheritance is a process in which one object acquires all the properties and behaviours of its parent object automatically. In such way, you can reuse, extend or modify the attributes and behaviours which are defined in other class.

In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

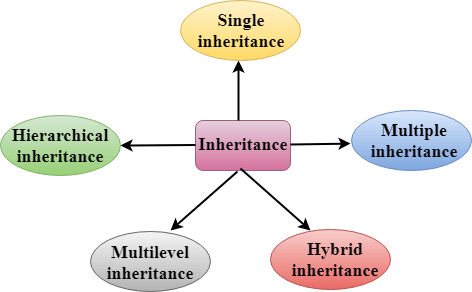
**Advantage of C++ Inheritance**

**Code reusability:** Now you can reuse the members of your parent class. So, there is no need to define the member again. So, less code is required in the class.

**Types Of Inheritance**

**C++ supports five types of inheritance:**

* Single inheritance
* Multiple inheritance
* Hierarchical inheritance
* Multilevel inheritance
* Hybrid inheritance



**Derived Classes**

A Derived class is defined as the class derived from the base class.

The Syntax of Derived class:

1. **class** derived\_class\_name :: visibility-mode base\_class\_name
2. {
3. // body of the derived class.
4. }

**Where,**

**derived\_class\_name:** It is the name of the derived class.

**visibility mode:** The visibility mode specifies whether the features of the base class are publicly inherited or privately inherited. It can be public or private.

**base\_class\_name:** It is the name of the base class.

|  |
| --- |
| / C++ program to demonstrate implementation of Inheritance    #include <iostream.h>  // Base class  **class** Parent  {  **public**:  **int** id\_p;  };    // Sub class inheriting from Base Class(Parent)  **class** Child : **public** Parent  {  **public**:  **int** id\_c;  };    // main function  **int** main()  {      Child obj1;        // An object of class child has all data members      // and member functions of class parent      obj1.id\_c = 7;      obj1.id\_p = 91;      cout << "Child id is: " <<  obj1.id\_c << '\n';      cout << "Parent id is: " <<  obj1.id\_p << '\n';    **return** 0;  } |

**Output**

Child id is 7

Parent id is 91

In the above program the ‘Child’ class is publicly inherited from the ‘Parent’ class so the public data members of the class ‘Parent’ will also be inherited by the class ‘Child’.

 **Modes of Inheritance**

1. **Public mode**: If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.
2. **Protected mode**: If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class.
3. **Private mode**: If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class.

**Note:**The private members in the base class cannot be directly accessed in the derived class, while protected members can be directly accessed. For example, Classes B, C and D all contain the variables x, y and z in below example. It is just question of access.

|  |
| --- |
| // C++ Implementation to show that a derived class  // doesn’t inherit access to private data members.  // However, it does inherit a full parent object.  **class** A  {  **public**:  **int** x;  **protected**:  **int** y;  **private**:  **int** z;  };    **class** B : **public** A  {      // x is public      // y is protected      // z is not accessible from B  };    **class** C : **protected** A  {      // x is protected      // y is protected      // z is not accessible from C  };    **class** D : **private** A    // 'private' is default for classes  {      // x is private      // y is private      // z is not accessible from D  }; |

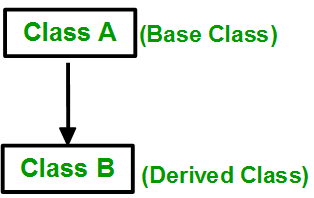
Note:

* In C++, the default mode of visibility is private.
* The private members of the base class are never inherited.

**Types of Inheritance in C++**

**1. Single Inheritance**:

In single inheritance, a class is allowed to inherit from only one class. i.e., one sub class is inherited by one base class only.



**Syntax**:

class subclass\_name : access\_mode base\_class

{

// body of subclass

};

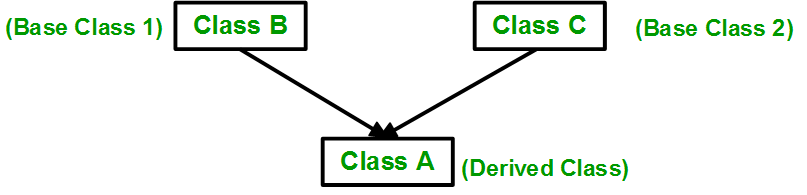
|  |
| --- |
| // C++ program to explain Single inheritance  #include<iostream>  **using** **namespace** std;   // base class  **class** Vehicle {  **public**:      Vehicle()      {        cout << "This is a Vehicle\n";      }  };    // sub class derived from a single base classes  **class** Car : **public** Vehicle {    };   // main function  **int** main()  {      // Creating object of sub class will      // invoke the constructor of base classes      Car obj;  **return** 0;  } |

**Output**

This is a Vehicle

**2. Multiple Inheritance:**

Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e one **sub class** is inherited from more than one **base classes**.



**Syntax**:

class subclass\_name : access\_mode base\_class1, access\_mode base\_class2, ....

{

// body of subclass

};

Here, the number of base classes will be separated by a comma (‘, ‘) and access mode for every base class must be specified.

|  |
| --- |
| // C++ program to explain multiple inheritance  #include<iostream>  **using** **namespace** std;   // first base class  **class** Vehicle {  **public**:      Vehicle()      {        cout << "This is a Vehicle\n";      }  };    // second base class  **class** FourWheeler {  **public**:      FourWheeler()      {        cout << "This is a 4 wheeler Vehicle\n";      }  };    // sub class derived from two base classes  **class** Car : **public** Vehicle, **public** FourWheeler {    };    // main function  **int** main()  {      // Creating object of sub class will      // invoke the constructor of base classes.      Car obj;  **return** 0;  } |

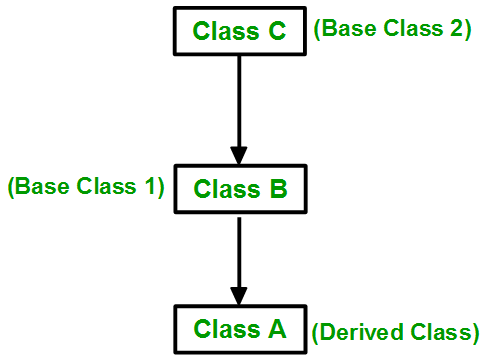
**Output**

This is a Vehicle

This is a 4 wheeler Vehicle

3.**Multilevel Inheritance**:

In this type of inheritance, a derived class is created from another derived class.



|  |
| --- |
| // C++ program to implement Multilevel Inheritance  #include<iostream>  **using** **namespace** std;   // base class  **class** Vehicle  {  **public**:      Vehicle()      {        cout << "This is a Vehicle\n";      }  };    // first sub\_class derived from class vehicle  **class** fourWheeler: **public** Vehicle  {  **public**:      fourWheeler()      {        cout << "Objects with 4 wheels are vehicles\n";      }  };  // sub class derived from the derived base class fourWheeler  **class** Car: **public** fourWheeler {  **public**:       Car()       {         cout << "Car has 4 Wheels\n";       }  };    // main function  **int** main()  {      // Creating object of sub class will      // invoke the constructor of base classes.      Car obj;  **return** 0;  } |

**Output**

This is a Vehicle

Objects with 4 wheels are vehicles

Car has 4 Wheels

**4. Hierarchical Inheritance**:

In this type of inheritance, more than one sub class is inherited from a single base class. i.e., more than one derived class is created from a single base class.



|  |
| --- |
| // C++ program to implement Hierarchical Inheritance  #include<iostream>  **using** **namespace** std;    // base class  **class** Vehicle  {  **public**:      Vehicle()      {        cout << "This is a Vehicle\n";      }  };    // first sub class  **class** Car: **public** Vehicle  {    };    // second sub class  **class** Bus: **public** Vehicle  {    };    // main function  **int** main()  {      // Creating object of sub class will      // invoke the constructor of base class.      Car obj1;      Bus obj2;  **return** 0;  } |

**Output**

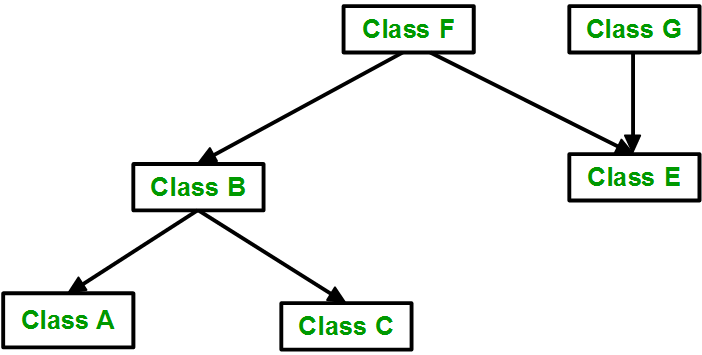
This is a Vehicle

This is a Vehicle

**5. Hybrid (Virtual) Inheritance**:

Hybrid Inheritance is implemented by combining more than one type of inheritance.

For example: Combining Hierarchical inheritance and Multiple Inheritance.   
Below image shows the combination of hierarchical and multiple inheritance:



|  |
| --- |
| // C++ program for Hybrid Inheritance    #include<iostream>  **using** **namespace** std;    // base class  **class** Vehicle  {  **public**:      Vehicle()      {        cout << "This is a Vehicle\n";      }  };    //base class  **class** Fare  {  **public**:      Fare()      {          cout << "Fare of Vehicle\n";      }  };    // first sub class  **class** Car : **public** Vehicle  {    };    // second sub class  **class** Bus : **public** Vehicle, **public** Fare  {    };   // main function  **int** main()  {      // Creating object of sub class will      // invoke the constructor of base class.      Bus obj2;  **return** 0;  } |

**Output**

This is a Vehicle

Fare of Vehicle