* **C++ Inheritance -**

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.

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

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

**Note:**

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

**C++ Single Inheritance**

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

**C++ single level inheritance example: Inheriting Fields**

**#include<iostream.h>**

**class Account**

**{**

**public:**

**float salary = 60000;**

**};**

**class Programmer: public Account**

**{**

**public:**

**float bonus = 5000;**

**};**

**void main()**

**{**

**Programmer p1;**

**cout<<"Salary: "<<p1.salary<<endl;**

**cout<<"Bonus: "<<p1.bonus<<endl;**

**}**

**C++ Single Level Inheritance Example: Inheriting Methods**

**#include<iostream.h>**

**class Animal**

**{**

**public:**

**void eat()**

**{**

**cout<<"Eating..."<<endl;**

**}**

**};**

**class Dog: public Animal**

**{**

**public:**

**void bark()**

**{**

**cout<<"Barking...";**

**}**

**};**

**void main()**

**{**

**Dog d1;**

**d1.eat();**

**d1.bark();**

**}**

**Another Example -**

**#include<iostream.h>**

**class A**

**{**

**int a = 4;**

**int b = 5;**

**public:**

**int mul()**

**{**

**int c = a\*b;**

**return c;**

**}**

**};**

**class B : private A**

**{**

**public:**

**void display()**

**{**

**int result = mul();**

**cout <<"Multiplication of a and b is : "<<result;**

**}**

**};**

**void main()**

**{**

**B b;**

**b.display();**

**}**

**How to make a Private Member Inheritable?**

The private member is not inheritable. If we modify the visibility mode by making it public, but this takes away the advantage of data hiding.C++ introduces a third visibility modifier, i.e., **protected**. The member which is declared as protected will be accessible to all the member functions within the class as well as the class immediately derived from it.

**-Visibility modes can be classified into three categories -**

* **Public**: When the member is declared as public, it is accessible to all the functions of the program.
* **Private**: When the member is declared as private, it is accessible within the class only.
* **Protected**: When the member is declared as protected, it is accessible within its own class as well as the class immediately derived from it.

**Visibility of Inherited Members**

|  |  |  |  |
| --- | --- | --- | --- |
| **Base class visibility** | **Derived class visibility** | | |
| **Public** | **Private** | **Protected** |
| Private | Not Inherited | Not Inherited | Not Inherited |
| Protected | Protected | Private | Protected |
| Public | Public | Private | Protected |

**C++ Multilevel Inheritance**

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

**#include<iostream.h>**

**class Animal**

**{**

**public:**

**void eat()**

**{**

**cout<<"Eating..."<<endl;**

**}**

**};**

**class Dog: public Animal**

**{**

**public:**

**void bark()**

**{**

**cout<<"Barking..."<<endl;**

**}**

**};**

**class BabyDog: public Dog**

**{**

**public:**

**void weep()**

**{**

**cout<<"Weeping...";**

**}**

**};**

**void main()**

**{**

**BabyDog d1;**

**d1.eat();**

**d1.bark();**

**d1.weep();**

**}**

**C++ Multiple Inheritance**

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

**#include<iostream.h>**

**class A**

**{**

**protected:**

**int a;**

**public:**

**void get\_a(int n)**

**{**

**a = n;**

**}**

**};**

**class B**

**{**

**protected:**

**int b;**

**public:**

**void get\_b(int n)**

**{**

**b = n;**

**}**

**};**

**class C : public A,public B**

**{**

**public:**

**void display()**

**{**

**cout << "The value of a is : " <<a;**

**cout << "The value of b is : " <<b;**

**cout<<"Addition of a and b is : "<<a+b;**

**}**

**};**

**void main()**

**{**

**C c;**

**c.get\_a(10);**

**c.get\_b(20);**

**c.display();**

**}**

**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.h>**

**class A**

**{**

**public:**

**void display()**

**{**

**cout << "Class A";**

**}**

**};**

**class B**

**{**

**public:**

**void display()**

**{**

**cout << "Class B";**

**}**

**};**

**class C : public A, public B**

**{**

**void view()**

**{**

**display();**

**}**

**};**

**void main()**

**{**

**C c;**

**c.display();**

**}**

**C++ Hybrid Inheritance**

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

**#include<iostream>**

**using namespace std;**

**class A**

**{**

**protected:**

**int a;**

**public:**

**void get\_a()**

**{**

**std::cout << "Enter the value of 'a' : " << std::endl;**

**cin>>a;**

**}**

**};**

**class B : public A**

**{**

**protected:**

**int b;**

**public:**

**void get\_b()**

**{**

**std::cout << "Enter the value of 'b' : " << std::endl;**

**cin>>b;**

**}**

**};**

**class C**

**{**

**protected:**

**int c;**

**public:**

**void get\_c()**

**{**

**std::cout << "Enter the value of c is : " << std::endl;**

**cin>>c;**

**}**

**};**

**class D : public B, public C**

**{**

**protected:**

**int d;**

**public:**

**void mul()**

**{**

**get\_a();**

**get\_b();**

**get\_c();**

**std::cout << "Multiplication of a,b,c is : " <<a\*b\*c<< std::endl;**

**}**

**};**

**int main()**

**{**

**D d;**

**d.mul();**

**return 0;**

**}**

**C++ Hierarchical Inheritance**

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

**#include<iostream.h>**

**class Shape**

**{**

**public:**

**int a,b;**

**void get\_data(int n,int m)**

**{**

**a = n;**

**b = m;**

**}**

**};**

**class Rectangle : public Shape**

**{**

**int result;**

**public:**

**int rect\_area()**

**{**

**result = a \* b;**

**return result;**

**}**

**};**

**class Triangle : public Shape**

**{**

**public:**

**int triangle\_area()**

**{**

**float result = 0.5\*a\*b;**

**return result;**

**}**

**};**

**void main()**

**{**

**Rectangle r;**

**Triangle t;**

**Int length,breadth,base,height,m;**

**float n;**

**cout<< "Enter the length and breadth of a rectangle: ";**

**cin>>length>>breadth;**

**r.get\_data(length,breadth);**

**m = r.rect\_area();**

**cout<<"Area of the rectangle is : " <<m;**

**cout<<"Enter the base and height of the triangle: ";**

**cin>>base>>height;**

**t.get\_data(base,height);**

**n = t.triangle\_area();**

**cout<<"Area of the triangle is : "<<n;**

**}**