Lab 07

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

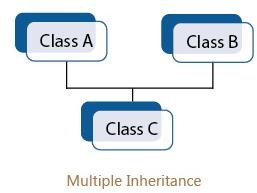
CL 1004 – Object Oriented Programming

BCY

Outline

Assa

* Inheritance
* Types of Inheritance: Multiple, Hierarchal , Hybrid
* Order of constructor and destructor calling.
* Examples
* Exercise

**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 class

## Syntax:

**class subclass\_name : access\_mode base\_class1, access\_mode base\_class2, {//body of subclass};**

class stud { protected:

int roll, m1, m2; public:

void get()

{ cout << "Enter the Roll No.: "; cin >> roll;

cout << "Enter the two highest marks: "; cin >> m1 >> m2; } };

class extracurriculam { protected:

int xm; public:

void getsm()

{ cout << "\nEnter the mark for Extra Curriculam Activities: "; cin >> xm; } }

class output : public stud, public extracurriculam { int tot, avg; public:

void display() {

tot = (m1 + m2 + xm); avg = tot / 3;

cout << "\n\n\tRoll No : "

<< roll << "\n\tTotal : " << tot;

cout << "\n\tAverage : " << avg; } }; int main() {

output O; O.get();

O.getsm();

O.display(); }

## 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.

|  |  |  |  |
| --- | --- | --- | --- |
| include <iostream> #include <string.h> using namespace std; class member {  char gender[10]; int age;  public:  void get()  {  cout << "Age: "; cin >> age; cout << "Gender: "; cin >>  gender;  }  void disp()  {  cout << "Age: " << age <<  endl;  cout << "Gender: " << gender << endl;  }}; | class stud : public member { char level[20];  public:  void getdata()  {  member::get();  cout << "Class: "; cin >> level;  }  void disp2()  {  member::disp();  cout << "Level: " << level  << endl;  }  }; | class staff : public member { float salary;  public:  void getdata()  {  member::get();  cout << "Salary: Rs."; cin  >> salary;  }  void disp3()  {  member::disp();  cout << "Salary: Rs." << salary << endl; | int main()  {  member M; staff S; stud s;  cout << "Student" << endl; cout << "Enter data" << endl; s.getdata();  cout << endl  << "Displaying data" << endl; s.disp();  cout << endl  << "Staff Data" << endl; cout << "Enter data" << endl; S.getdata();  cout << endl  << "Displaying data" << endl; S.disp();  } |

Output

Staff Data Enter data Age: 12 Gender: male

Salary: Rs.100000 Displaying data Age: 12

Gender: male

Student Enter data Age: 12

Gender: Female Class: 10 Displaying data Age: 12 Gender: Female

## Hybrid (Virtual) Inheritance

* + Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.

This is a Vehicle Fare of Vehicle

// base class class Vehicle

{

public:

Vehicle()

{

cout << "This is a Vehicle"

<< endl;

}

};

//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;

}

### Order of constructor and Destructor call

#include <iostream> using namespace std;

// base class class Parent

{

public:

// base class constructor

Parent()

{

cout << "Inside base class" << endl;

}

// sub class

class Child : public Parent {

public:

//sub class constructor Child()

{ cout << "Inside sub

class" << endl;

}

};

|  |  |
| --- | --- |
|  | Output |
| // main function int main() { |  |
| Inside base class |
| // creating object of sub class  Child obj;  return 0;  } | Inside sub class |
|  |

**Concept:** Calling parameterized constructor of base class in derived class constructor!

* + To call the parameterized constructor of base class when derived class’s parameterized constructor is called, you have to explicitly specify the base class’s parameterized constructor in derived class

**Concept:** Calling parameterized constructor of base class in derived class constructor!

### Program as Example

class Base

{

int x; public:

// parameterized constructor Base(int i)

{

x = i;

cout << "Base Parameterized Constructor\n";

}

};

class Derived : public Base{ int y;

public:

// parameterized constructor Derived(int j):Base(j)

{

y = j;

cout << "Derived Parameterized

Constructor\n";

}

};

int main()

{

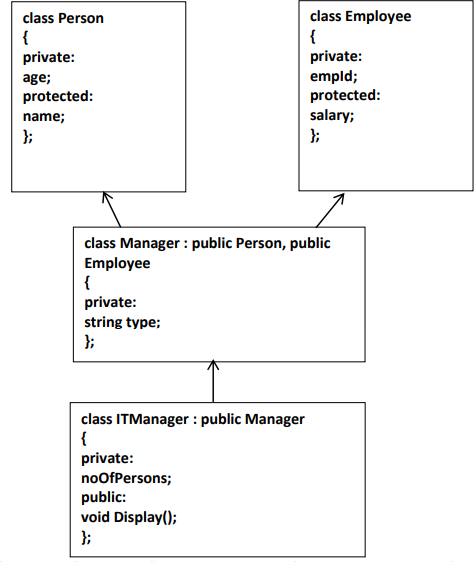
Derived d(10) ;

}

**Output**

Base Parameterized Constructor Derived Parameterized Constructor

Activity

1. Make a class named Fruit with a data member to calculate the number of fruits in a basket. Create two other class named Apples and Mangoes to calculate the number of apples and mangoes in the basket. Print the number of fruits of each type and the total number of fruits in the basket.
2. Create two classes named Mammals and MarineAnimals. Create another class named BlueWhale which inherits both the above classes. Now, create a function in each of these classes which prints "I am mammal", "I am a marine animal" and "I belong to both the categories: Mammals as well as Marine Animals" respectively. Now, create an object for each of the above class and try calling
   * function of Mammals by the object of Mammal
   * function of MarineAnimal by the object of MarineAnimal
   * function of BlueWhale by the object of BlueWhale
   * function of each of its parent by the object of BlueWhale
3. We want to store the information of different vehicles. Create a class named Vehicle with two data member named mileage and price. Create its two subclasses  
   \*Car with data members to store ownership cost, warranty (by years), seating capacity and fuel type (diesel or petrol).  
   \*Bike with data members to store the number of cylinders, number of gears, cooling type(air, liquid or oil), wheel type(alloys or spokes) and fuel tank size(in inches)  
   Make another two subclasses Audi and Ford of Car, each having a data member to store the model type. Next, make two subclasses Bajaj and TVS, each having a data member to store the make-type.  
   Now, store and print the information of an Audi and a Ford car (i.e. model type, ownership cost, warranty, seating capacity, fuel type, mileage and price.) Do the same for a Bajaj and a TVS bike.
4. Implement the following scenario: