**Labsheet-2: Classes And Objects**

***Objectives***

* Recognize how to specify the class.
* Define the member functions.
* Explain the creation of class objects.
* Access the class members.
* Discuss access specifier.

***Introduction***

Classes and objects are the core of object-oriented programming. Writing program in C++ essentially means writing classes and creating object from them.

**Class**

* A class is a template that specify different aspect of object it models.
* No physical existence.
* Occupies no memory
* Define data attribute and/or methods that constitute class.

**Object**

* Object is instance of class.
* Has physical existence and occupy memory.
* We can create as many objects as we like once we have defined class.

The **General Form** of class declaration is:

class ClassName{

private:   
 variable declarations; and/or function declarations;

public:

variable declarations; and/or function declarations;

};

In above syntax, **class** is C++ keyword and ClassName is name of the class. The body of class is enclosed within a pair of braces and terminated by semicolon. The class body contains the declaration of variables and functions.

Keyword **private** and **public** within class body specifies **visibility mode** of variables and functions I.e they specify which of the members are private and which of them are public. Using private declaration **data** **hiding** is possible. By default all members of the class are private. Conventionally, the data within a class are made private and functions are made public. C++ also provides **protected** mode, which is used in inheritance.

**Creating objects**

For a class Rational, any object of type Rational can be created as:

Rational a, b; // a and b are objects of type Rational.

**Accessing class members**

Members of objects can be accessed using dot(.) operator as:

a. num;

b.print()

double value = b.convert()

**Defining methods outside class.**

Method can be defined inside or outside of the class. If a method is defined outside the body of class, we need to use scope resolution operator (**::**).

// Syntax to define methods outside class

void ClassName :: function\_name(parameters)

{

// Function Body

}

**Assignments:**

1. **Write a complete definition of the class Rational with its methods defined outside.**

**Answer:**

|  |
| --- |
| #include<iostream> #include<conio.h> using namespace std;  class Rational{  int num, den;  public:  void assign();  void display();  };  void Rational :: assign(){  cout << "Enter numerator and denominator: ";  cin >> num >> den;  }  void Rational :: display(){  cout << "Rational number = " << num << "/" << den << endl;  }  int main(){  Rational R;  R.assign();  R.display();  getch();  return 0;  } |

**Output:**

Enter numerator and denominator:: **22** **7**

Rational number = 22/7

1. **Write a program to implement the class. The program should**
   1. **Define Rational object x.**
   2. **Assign 22/7 to x.**
   3. **Print the object’s value as rational number.**
   4. **Print the same rational number as a real number.**
   5. **Invert x, such that X is 1/x.**

|  |
| --- |
| #include<iostream> #include<conio.h> using namespace std;  class Rational{  int num, den;  public:  void assign(int , int );  void display();  void swap(Rational & );  };  void Rational :: assign(int n, int d){  num = n; den = d;  }  void Rational :: display(){  cout << "Rational number = " << num << "/" << den  << " = " << (num \* 1.0 / den) << endl; }  void Rational :: swap(Rational & r){  int temp;  temp = r.num;  r.num = r.den;  r.den = temp;  }  int main(){  Rational x;  x.assign(22, 7);  x.display();  x.swap(x);  cout << "After Inverse" << endl;  x.display();  getch();  return 0;  } |

**Possible Output:**

Rational Number = 22/7 = 3.14286

After Inverse

Rational Number = 7/22 = 0.318182

**g. Check if you can print num and den of x from main. Discuss the reason behind your observation.**

**Ans:** No, We cannot print num and den of x from main. Say, we want to print num by cout << num; then we get following error message:

|error: ‘num’ was not declared in this scope|

Since, num and den are the data attributes of private section they are not visible to outside. If you have members under a Private access specifier then those members are only accessible from within the class. No outside Access is allowed. This is a feature of C++ known as **data hiding.**

**h. If you were not able to access num and den of x from outside. What should you do so that you can access them from anywhere?**

**Ans:** To access the private sections data attribute we can use the member function of the public section. In the above example also we use a public member function display() to print the data attribute of the private sections.

1. **Create a class Point that represents a three dimensional coordinate system. Each object of Point should have coordinates (x,y,z) and methods to assign coordinates to the point objects. Add a method to calculate the distance from origin and to print the point in the form of (x,y,z)**

**Answer:**

|  |
| --- |
| #include<iostream> #include<conio.h> #include<math.h>  using namespace std;  class Point{  float x, y, z;  public:   void assign(float a, float b, float c)  {x =a; y = b; z = c;}  void display();  float distance(); };  void Point :: display(){  cout << "(x, y, z) = " << "(" <<x<<","<<y <<","<<z<<")"<<endl;  }  float Point :: distance(){  return sqrt(x\*x + y\*y + z\*z);  }  int main(){  Point P;  P.assign(1, 0, 1);  P.display();  cout << "Distance from origin = " << P.distance() << endl;  getch();  return 0;  } |

**Output:**

(x, y, z) = (1, 0, 1)

Distance from origin = 1.41421

1. **Design a class called Person that contains appropriate members for storing name, age, gender and telephone number. Write member functions that can read and display these data.**

|  |
| --- |
| #include<iostream>  #include<conio.h> #include<string.h> using namespace std;  class Person{  char name[33], gender[6];  int age; long telephone;  public:  void assign(char \* n, int a, char \* g, long t){  strcpy(name, n);  strcpy(gender, g);  age = a; telephone = t;  }  void display();  };  void Person :: display()  {  cout << "Name: " << name << "\nAge: " << age  << "\nTelephone: " << telephone  << "\nGender: " << gender << endl;  }  int main()  { Person P1;  P1.assign("Ramesh", 12, "Male", 9847585858);  P1.display();  //getch();  return 0;  } |

Name : Ramesh

Age : 12

Gender : Male

Telephone : 9847585858

1. **Write a program to represent a Circle. Include member functions to perform the following task:**
   1. **Calculate area of the Circle**
   2. **Calculate perimeter of the circle**

|  |
| --- |
| #include<iostream>  #include<conio.h>  using namespace std;  class Circle{  float radius;  public: void assign();  float area();  float perimeter();  };  void Circle :: assign(){  cout << "Enter radius = ";  cin >> radius;  }  float Circle :: area(){  return 3.14 \* radius \* radius;  }  float Circle :: perimeter(){  return 2 \* 3.14 \* radius;  }  int main(){  Circle C1;  C1.assign();  cout << "Area = " << C1.area() << endl;  cout << "Perimeter = " << C1.perimeter() << endl;  getch(); return 0;  } |

**Possible Output:**

Enter Radius = 10

Area = 314

Perimeter = 62.8

**Conclusion**:

A **class** represents a group of similar objects. A class in C++ binds data and associated function together. It makes a data type using which objects of this type can be created. Classes can represent real world object which have characteristics and and associated behavior.

While declaring class four attributes are declared:

1. Data Members.

2. Member Functions

3. Program Access Levels(private, public, private)

4. Class Tag Name.

While defining a class, its member function can be defined within class or outside class. If we define member function outside class then we need **scope resolution** operator (::).

Members of objects can be accessed using **dot**(.) operator. The **private** and **protected** members can be accessed with in the class by the member function only. The member function of a class is also called a **method.**

C++ allows us to have arrays of objects. We may pass object as a function arguments. A function can also return an object.

Thus, In this chapter we were able to explore various aspects of classes and objects. We learn how to specify a class and how to create the objects using that class. We also learn that we can create as much object as we like once the class is defined. We also learn to access data members either in public or private sections.