Academic Year: 2020

Semester: 2nd

Course Code: CS-241L

Course Title: Object Oriented Programming

CS-241L Object Oriented Programming Lab 04

Type of Lab: Open Ended **Weightage: 5%**

CLO: CLO1 + CLO2

Student understand the concept of constructor, object as parameter and access modifier	**Cognitive/Understanding**	CLO1, CLO2	Rubric A

Rubric A: Cognitive Domain

Evaluation Method: GA shall evaluate the students for Question according to following rubrics.

CLO	**0**	**1**	**2**	**3**	**4**	**5**
CL01 CLO2	Unable to understand and implement	Student understand the constructor concept	Implement constructor and destructor	Implement access specifier concept	Understand and implemented half problem sets	Understand and implemented complete problem sets

Lab 4

BS-Computer Science

Object Oriented Programming

Target: Write different type of constructors, Constructor Overloading, Destructor, Objects as function parameters Implement Access Modifiers

Constructors: Constructors are special class members which are called by the compiler every time an object of that class is instantiated. Constructors have the same name as the class and may be defined inside or outside the class definition. There are 3 types of constructors:

- **Default Constructer**
- **Parametrized Constructor**
- **Copy Constructor**

  Default Constructors: Default constructor is the constructor which doesn't take any argument. It has no parameters.

  Parameterized Constructors: It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created.

  Copy Constructor: A copy constructor is a member function which initializes an object using another object of the same class.

Default Constructors: Default constructor is the constructor which doesn't take any argument. It has no parameters.

Example:

```
class default_construct
{
   public:
   int id;

   //Default Constructor
   default_construct()
```

```
  {
    cout << "Default Constructor called" << endl;
    id=-1;
  }
};
```

Parameterized Constructors: It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created.

Problem Set:

With the help of comments in the code, explain it and modify it for 2 parameters in the constructor. Write down the output of the code.

```
#include <stdio.h>
class param_construct
{
  public:
  int id;

  //Parametrized Constructor
  Param_construct(int x)
  {
    cout << "Parametrized Constructor called" << endl;
    id=x;
  }
};
int main() {

  // obj1 will call Parameterized Constructor
  Param_constuct obj1(20);

  cout << "Param id is: " <<obj1.id << endl;
  return 0;
}
```

Copy Constructor: A copy constructor is a member function which initializes an object using another object of the same class.

Problem Set:

Elaborate the use of copy constructor in the code given below, write down the usage and execution dry run on the paper. Write down the output of the code.

```
#include<iostream>

class Point
{
private:
   int x, y;
public:
   Point(int x1, int y1) { x = x1; y = y1; }

   // Copy constructor
   Point(const Point &p2) {x = p2.x; y = p2.y; }

   int getX()          {  return x; }
   int getY()          {  return y; }
};

int main()
{
   Point p1(10, 15); // Normal constructor is called here
   Point p2 = p1; // Copy constructor is called here

   // Let us access values assigned by constructors
   cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();
   cout << "\np2.x = " << p2.getX() << ", p2.y = " << p2.getY();

   return 0;
}
```

When is copy constructor called?

In C++, a Copy Constructor may be called in following cases:

- When an object of the class is returned by value.
- When an object of the class is passed (to a function) by value as an argument.
- When an object is constructed based on another object of the same class.
- When the compiler generates a temporary object.

Constructor Overloading:

The process of declaring multiple constructors with same name but different with parameters. The Constructors with same names can be different in three ways

- Number of parameters
- Type of parameters
- Sequence of parameters

Destructor:

Destructor is a special class function which destroys the object as soon as the scope of object ends (destroys) when program is being terminated and all objects destroyed from memory. The destructor is called automatically by the compiler when the object goes out of scope. Press Alt+F5 to view the output after terminating the program.

The syntax for destructor is same as that for the constructor with no return type and cannot accept parameters and name is same as class with a **tilde** ~ sign as prefix to it.

```
~A()
{
  // statement
}
```

Objects as Functions Parameters:

Objects of class can also be passed as parameters in member functions. Same method of passing as in user defined functions.

Problem Set:

- Define a class of Travel with the following specifications:

Class of Travel

Private Members	Type
Km (kilometers), hr (hours)	Integer
Public Member:	
Travel() (default constructor initializes both members to 0).	none
Input() **Detail:** takes inputs from user	void
displaydata() **Detail:** Function to display the data members on the screen	Void
Sum(object as parameter) **Detail:** that takes an object as parameter of type as class and adds kilometers and hours of calling object and the parameter and displays values.	void

Returning Objects from member Functions:

Same as in simple method and return type should be same as class name and it returns an object.

/////////Do yourself

- Define a class of Time with the following specifications:

~~Class of Batsman~~

Private Members	Type
hrs(hours), min(minutes)	Integer
Public Member:	
Time() (default constructor initializes both members to 0).	none
TInput() **Detail:** takes inputs as minutes and hours from user	void

displaydataT() **Detail:** Function to display the data members on the screen	Void
time sum(time) **Detail:** that takes an object as parameter of type as class and adds to sum two-time object & return time**)** .	void

Access Modifiers: These are used to implement important feature of OOP known as **Data Hiding.**
Consider a real life example: What happens when a driver applies brakes? The car stops. The driver only knows that to stop the car, he needs to apply the brakes. He is unaware of how actually the car stops. That is how the engine stops working or the internal implementation on the engine side. This is what data hiding is. Access modifiers or Access Specifiers in a class are used to set the accessibility of the class members. That is, it sets some restrictions on the class members not to get directly accessed by the outside functions.

There are 3 types of access modifiers available in C++:

- **Public**
- **Private**
- **Protected**

Public: All the class members declared under public will be available to everyone. The data members and member functions declared public can be accessed by other classes too. The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class.

Private: The class members declared as private can be accessed only by the functions inside the class. They are not allowed to be accessed directly by any object or function outside the class. Only the member functions or the friend functions are allowed to access the private data members of a class.

Protected: Protected access modifier is similar to that of private access modifiers, the difference is that the class member declared as Protected are inaccessible outside the class but they can be accessed by any subclass (derived class) of that class. [inheritance comes here, will be explained with that]

Note: If we do not specify any access modifiers for the members inside the class then by default the access modifier for the members will be **Private**.

Problem Set:

Given the code for public access modifier, demonstrate the scope of member functions and data members. Write down the output of the program.

```
// C++ program to demonstrate public access modifier

#include<iostream>

// class definition
class Circle
{
   public:
      double radius;

      double  compute_area()
      {
         return 3.14*radius*radius;
      }

};

// main function
int main()
{
   Circle obj;
```

```
    // accessing public datamember outside class
    obj.radius = 5.5;

    cout << "Radius is:" << obj.radius << "\n";
    cout << "Area is:" << obj.compute_area();
    return 0;
}
```

Problem Set:

Given the code for private access modifier, demonstrate the scope of member functions and data members. Write down the output of the program. If there is any error, write down the reason of that error.

```
// C++ program to demonstrate private access modifier

#include<iostream.h>

class Circle
{
    // private data member
    private:
        double radius;

    // public member function
    public:
        double  compute_area()
        {   // member function can access private
            // data member radius
            return 3.14*radius*radius;
        }

};

// main function
int main()
```

```
{
   // creating object of the class
   Circle obj;

   // trying to access private data member
   // directly outside the class
   obj.radius = 1.5;

   cout << "Area is:" << obj.compute_area();
   return 0;
}
```

The output of above program will be a compile time error because we are not allowed to access the private data members of a class directly outside the class.