



Object Oriented Programming (CS02203)

Lab Report

Name: Noor Baho
Registration #: CSU-F12-116
Lab Report #: 05
Dated: 18-11-2018
Submitted To: Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 5

Object Oriented Paradigm, Classes and Objects

Objective

Objective of this lab session is to reinforce programming concepts regarding to class, object constructor and destructors.

Software Tool

- OS Windows 10
- Editor DEV C++
- Language C++

1 Theory

1.1 Classes

Classes are collections of data related to a single object type along with functions to access and manipulate the data. The functions usually are the only way to modify and access the variables. This might seem silly at first, but the idea to make programs more modular - the principle itself is called "encapsulation". The key idea is that the outside world doesn't need to know exactly what data is stored inside the class, it just needs to know which functions it can use to access that data. This allows the implementation to change more easily because nobody should have to rely on it except the class itself.

Description

The mechanism that allows you to combine data and the function in a single unit is called a class. Once a class is defined, you can declare variables of that type. A class variable is called object or instance. In other words, a class would be the data type, and an object would be the variable. Classes are generally declared using the keyword `class`, with the following format:

```
class class_name
```

```

{
    private :
    members1;
    protected :
    members2;
    public :
    members3;
};

```

Where class name is a valid identifier for the class. The body of the declaration can contain members, that can be either data or function declarations, The members of a class are classified into three categories: private, public, and protected. Private, protected, and public are reserved words and are called member access specifiers. These specifiers modify the access rights that the members following them acquire.

Private members of a class are accessible only from within other members of the same class. You cannot access it outside of the class.

Protected members are accessible from members of their same class and also from members of their derived classes.

Public members are accessible from anywhere where the object is visible. By default, all members of a class declared with the class keyword have private access for all its members. Therefore, any member that is declared before one other class specifier automatically has private access.

Here is a complete example:

```

class student
{
    private :
    int rollno;
    float marks;
    public :
    void getdata()
    {
        cout<<"Enter Roll Number : ";
        cin>>rollno;
        cout<<"Enter Marks : ";
        cin>>marks;
    }
}

```

```

        void displaydata()
        {
            cout<<"Roll number : "<<rollno<<"\n Marks : "<<marks;
        }
};

```

1.2 Object Declaration

Once a class is defined, you can declare objects of that type. The syntax for declaring a object is the same as that for declaring any other variable. The following statements declare two objects of type student:

```
student st1, st2;
```

Accessing Class Members

Once an object of a class is declared, it can access the public members of the class.

```
st1.getdata();
```

Defining Member Function of Class

You can define Functions inside the class as shown in above example. Member functions defined inside a class this way are created as inline functions by default. It is also possible to declare a function within a class but define it elsewhere. Functions defined outside the class are not normally inline.

When we define a function outside the class we cannot reference them (directly) outside of the class. In order to reference these, we use the scope resolution operator, :: (double colon). In this example, we are defining function getdata outside the class

```

void student :: getdata()
{
    cout<<"Enter Roll Number : ";
    cin>>rollno;
    cout<<"Enter Marks : ";
    cin>>marks;
}

```

The following program demonstrates the general feature of classes. Member function initdata() is defined inside the class. Member functions getdata()

and showdata() defined outside the class.

```
class student //specify a class
{
private :
int rollno; //class data members
float marks;
public:
void initdata(int r, int m)
{
rollno=r;
marks=m;
}
void getdata(); //member function to get data from user
void showdata();// member function to show data
};
void student :: getdata()
{
cout<<"Enter Roll Number : ";
cin>>rollno;
cout<<"Enter Marks : ";
cin>>marks;
}
void student :: showdata() // Display the data
{
cout<<"Roll number : "<<rollno<<"\nMarks : "<<marks;
}
int main()
{
student st1, st2; //define two objects of class student
st1.initdata(5,78); //call member function to initialize
st1.showdata();

st2.getdata(); //call member function to input data
st2.showdata(); //call member function to display data
return 0;
}
```

1.3 Constructor

Constructors are special class functions which performs initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors initialize values to object members after storage is allocated to the object.

```
class A
{
int x;
public:
A(); //Constructor
};
```

While defining a constructor you must remember that the name of constructor will be same as the name of the class, and constructors never have return type.

Constructors can be defined either inside the class definition or outside class definition using class name and scope resolution :: operator.

```
class A
{
int i;
public:
A(); //Constructor declared
};
A::A() // Constructor definition
{
i=1;
}
```

Types of Constructors

Constructors are of three types:

- **Default Constructor**
- **Parametrized Constructor**

- **Copy Constructor**

Default Constructor

Default constructor is the constructor which doesn't take any argument. It has no parameter.

Syntax:

```
class_name ()  
{ Constructor Definition }  
Example :  
class Cube  
{  
int side;  
public:  
Cube()  
{  
side=10;  
}  
};  
int main()  
{  
Cube c;  
cout << c.side;  
}
```

Output: 10

In this case, as soon as the object is created the constructor is called which initializes its data members.

A default constructor is so important for initialization of object members, that even if we do not define a constructor explicitly, the compiler will provide a default constructor implicitly.

```
class Cube  
{  
  
int side;  
};  
int main()  
{
```

```
Cube c;
cout << c.side;
}
```

Output: 0

In this case, default constructor provided by the compiler will be called which will initialize the object data members to default value that will be 0 in this case.

Parameterized Constructor

These are the constructors with parameter. Using this Constructor you can provide different values to data members of different objects, by passing the appropriate values as argument.

Example :

```
class Cube
{
int side;
public:
Cube(int x)
{
side=x;
}
};
int main()
{
Cube c1(10);
Cube c2(20);
Cube c3(30);
cout << c1.side;

cout << c2.side;
cout << c3.side;
}
```

OUTPUT : 10 20 30

By using parameterized constructor in above case, we have initialized 3 objects with user defined values. We can have any number of parameters in a

constructor.

Copy Constructor

These are special type of Constructors which takes an object as argument, and is used to copy values of data members of one object into other object. We will study copy constructors in detail later.

Constructor Overloading

Just like other member functions, constructors can also be overloaded. In fact when you have both default and parameterized constructors defined in your class you are having Overloaded Constructors, one with no parameter and other with parameter. You can have any number of Constructors in a class that differ in parameter list.

```
class Student
{
int rollno;
string name;
public:
Student(int x)
{
rollno=x;
name="None";
}
Student(int x, string str)
{
rollno=x ;
name=str ;
}
};
int main()

{
Student A(10);
Student B(11,"Ram");
}
```

In above case we have defined two constructors with different parameters, hence overloading the constructors. One more important thing, if you define any constructor explicitly, then the compiler will not provide default constructor and you will have to define it yourself. In the above case if we

write Student S; in main(), it will lead to a compile time error, because we haven't defined default constructor, and compiler will not provide its default constructor because we have defined other parameterized constructors.

1.4 Destructor

Destructor is a special class function which destroys the object as soon as the scope of object ends. The destructor is called automatically by the compiler when the object goes out of scope. The syntax for destructor is same as that for the constructor, the class name is used for the name of destructor, with a tilde sign as prefix to it.

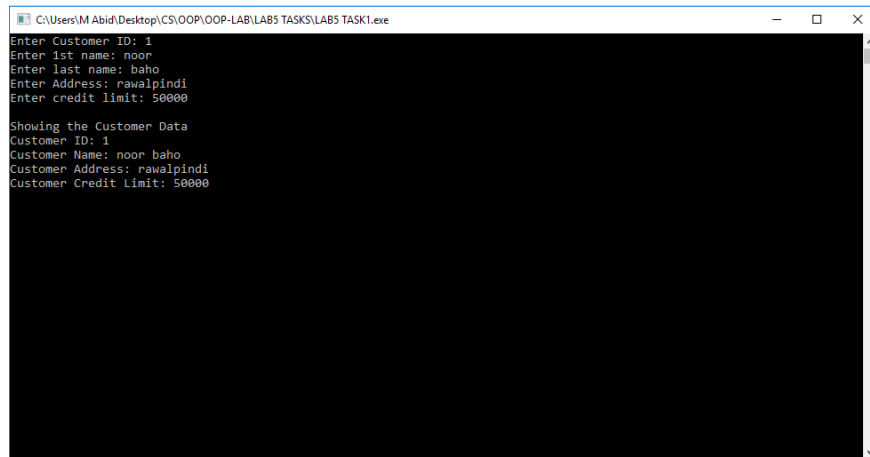
```
class A
{
public:
~A();
};
```

Destructors will never have any arguments.

How constructor and destructor is called

```
class A
{
A()
{
cout << "Constructor called";
}
~A()
{
cout << "Destructor called";
}
};
int main()
{
A obj1; // Constructor Called
int x=1
```

```
if(x)
{
A obj2; // Constructor Called
} // Destructor Called for obj2
} // Destructor called for obj1
```



```
C:\Users\Abid\Desktop\CS\OOP\OOP-LAB\LAB5 TASKS\LAB5 TASK1.exe
Enter Customer ID: 1
Enter 1st name: noor
Enter last name: baho
Enter Address: rawalpindi
Enter credit limit: 50000

Showing the Customer Data
Customer ID: 1
Customer Name: noor baho
Customer Address: rawalpindi
Customer Credit Limit: 50000
```

Figure 1: Record of Customer

2 Task

2.1 Procedure: Task 1

Define a class Customer that holds private fields for a customer's ID, first name, last name, address and credit limit.

Define functions that set the fields of customer. For example setCustomerID().

Define functions that show the fields of customer. For example getCustomerID().

Use constructor to set the default values to each of the field.

Overload at least six constructor of the customer class.

Define a function that takes input from user and set the all fields of customer data.

Define a function that displays the entire customer's data.

Solution:

```
#include<iostream>
#include<conio.h>
//customer class set members and get memebers
using namespace std;
class Customer{
    private:
```

```

        int id, credit_limit;
        string f_name, l_name;
        string address;

public:
    Customer()                //constructor
    {
        id=1;
        credit_limit=0;
        f_name="firstname";
        l_name="lastname";
        address="Pakistan";
    }
    void set_id()
    {
        cout<<"Enter Customer ID: ";
        cin>>id;
    }
    void set_name()
    {
        cout<<"Enter 1st name: ";
        cin>>f_name;
        cout<<"Enter last name: ";
        cin>>l_name;
    }
    void set_address()
    {
        cout<<"Enter Address: ";
        cin>>address;
    }
    void set_credit_limit()
    {
        cout<<"Enter credit limit: ";
        cin>>credit_limit;
    }

    void setCustomer()        //general input function
    {
        set_id();
        set_name();

```

```

        set_address ();
        set_credit_limit ();
    }

    void get_id ()
    {
        cout<<"Customer ID: "<<id<<endl;
    }
    void get_name ()
    {
        cout<<"Customer Name: "<<f_name;
        cout<<" "<<l_name<<endl;
    }
    void get_address ()
    {
        cout<<"Customer Address: "<<address;
        cout<<endl;
    }
    void get_credit_limit ()
    {
        cout<<"Customer Credit Limit: "<<credit_limit;
        cout<<endl;
    }

    void getCustomer ()
    {
        cout<<endl<<"Showing the Customer Data"<<endl;
        get_id ();
        get_name ();
        get_address ();
        get_credit_limit ();
    }

};

int main()
{
    Customer cust1;

    cust1.setCustomer ();

```

```
        cust1.getCustomer();  
        getch();  
        return 0;  
    }
```

Explanation: In this program, we created a class with name customer which have all member variables private and operations (functions) public. Also a constructor to initialize data. To access the member functions in main function, we created an object of customer class named cust1 through which we first input value from user and then shown that value to the screen.

```

C:\Users\Manish\Desktop\CO-LOOP\OOP-LAB\LAB TASKS\LAB TASK2.exe
ENTER COUNTRY DATA:
Enter name of Country: pakistan
Enter Longitude: 5678
Enter Latitude: 5678
Enter area of country: 23456
Enter population of country: 456789
Enter Capital of country: islamabad

ENTER COUNTRY DATA:
Enter name of Country: india
Enter Longitude: 876
Enter Latitude: 345
Enter area of country: 543
Enter population of country: 655442
Enter Capital of country: delhi

ENTER COUNTRY DATA:
Enter name of Country: iran
Enter Longitude: 567
Enter Latitude: 5432
Enter area of country: 543422
Enter population of country: 65443
Enter Capital of country: tehran

SHOWING DATA OF COUNTRY: Name of Country: pakistan
Longitude of Country: 5678
Latitude of Country: 5678
Area of Country: 23456
Capital of Country: islamabad
Name of Country: india
Longitude of Country: 876
Latitude of Country: 345
Area of Country: 543
Capital of Country: delhi
Name of Country: iran
Longitude of Country: 567
Latitude of Country: 5432
Area of Country: 543422
Capital of Country: tehran

```

Figure 2: Temperature Converter

2.2 Procedure: Task 2

Create a class Country. Country is identified by its name, location on earth (which means the longitude and latitude values), area, population and capital. All the data members should be private. User should be able to set the value of all of its parameters either at the time of instance creation or later on be able to individually change the values of all these identifiers whenever user want to. Calling member function DisplayAllInfo() should display all the information pertaining to a country instance. User should also be able to individually call functions such as DisplayName(), DisplayCapital etc.

To test: Ask the user to enter 3 countries and all the related info. Display the info once entered.

Solution:

```

#include<iostream>
#include<conio.h>

using namespace std;

class Country{
private:
    string c_name, capital;
    int c_id;

```



```

        long int longitude , latitude , population ;
        double area ;

public :

        void set_name ()
        {
                cout<<"Enter name of Country: ";
                cin>>c_name;
        }
        void set_location ()
        {
                cout<<"Enter Longitude: ";
                cin>>longitude;
                cout<<"Enter Latitude: ";
                cin>>latitude;
        }
        void set_area ()
        {
                cout<<"Enter area of country: ";
                cin>>area;
        }
        void set_population ()
        {
                cout<<"Enter population of country: ";
                cin>>population;
        }
        void set_capital ()
        {
                cout<<"Enter Capital of country: ";
                cin>>capital;
        }
        void SetInfo ()
        {
                set_name ();
                set_location ();
                set_area ();
                set_population ();
                set_capital ();
        }

```

```

void Display_name()
{
    cout<<"Name of Country: "<<c_name<<endl;
}
void Display_location()
{
    cout<<"Logitude of Country: "<<longitude<<endl;
    cout<<"Latitude of Country: "<<latitude<<endl;
}
void Display_area()
{
    cout<<"Area of Country: "<<area<<endl;
}
void Display_capital()
{
    cout<<"Capital of Country: "<<capital<<endl;
}
void DisplayAllInfo()
{
    Display_name();
    Display_location();
    Display_area();
    Display_capital();
}

};

int main()
{
    Country c[3];
    int changechoice, allfieldchoice, fieldchoice, countrychoice;

    for(int x=0; x<=2; x++) //input three country data
    {
        cout<<"ENTER COUNTRY DATA:"<<endl;
        c[x].SetInfo();
        cout<<endl<<endl;
    }
}

```

```

}

cout<<endl<<endl<<"SHOWING DATA OF COUNTRIES"<<endl;
for (int x=0; x<=2; x++)
{

    c[x].DisplayAllInfo();
    cout<<endl;
}

cout<<"Do you want to change any Country info?";
cout<<" If yes then enter 1, if no then enter 0: ";
cin>>changechoice;
if (changechoice==1)
{
    cout<<"For which country you want to change";
    cout<<" information? Enter the id: ";
    cin>>countrychoice;
    cout<<"To change all fields press 1, To change";
    cout<<" any single field press 2: ";
    cin>>allfieldchoice;
    countrychoice-=1;
    if (allfieldchoice==1)
    {
        c[countrychoice].SetInfo();
        cout<<endl<<endl<<"Showiong Updated Data:";
        cout<<endl;
        c[countrychoice].DisplayAllInfo();
    }
    else if (allfieldchoice==2)
    {
        cout<<"Which field you want to change?";
        cout<<" Press 1 for Country name,";
        cout<<" 2 for location,";
        cout<<" 3 for area, 4 for population";
        cout<<" and 5 for capital : ";
        cin>>fieldchoice;
        switch (fieldchoice)
        {

```

```

        case '1':
            c[countrychoice].set_name();
            cout<<endl<<endl<<"Showing you";
            cout<<" Updated Record:"<<endl;
            c[countrychoice].DisplayAllInfo();
            break;
        case '2':
            c[countrychoice].set_location();
            cout<<endl<<endl<<"Showing you";
            cout<<" Updated Record:"<<endl;
            c[countrychoice].DisplayAllInfo();
            break;
        case '3':
            c[countrychoice].set_area();
            cout<<endl<<endl<<"Showing you";
            cout<<" Updated Record:"<<endl;
            c[countrychoice].DisplayAllInfo();
            break;
        case '4':
            c[countrychoice].set_name();
            cout<<endl<<endl<<"Showing you";
            cout<<" Updated Record:"<<endl;
            c[countrychoice].DisplayAllInfo();
            break;
        case '5':
            c[countrychoice].set_name();
            cout<<endl<<endl<<"Showing you ";
            cout<<"Updated Record:"<<endl;
            c[countrychoice].DisplayAllInfo();
            break;
        case '6':
            c[countrychoice].set_name();
            cout<<endl<<endl<<"Showing you";
            cout<<" Updated Record:"<<endl;
            c[countrychoice].DisplayAllInfo();
            break;
    }
}
return 0;

```

}

Explanation: For desired Execution, created a class named country which has all member variables private but member functions public. For each attribute there is a function to take input from user and a general function to give output to screen of all attributes. Now in main function, we created an object array of class country which have index 3 as we are required to input 3 countries. Taking input from user for all three countries then showing all taken input back to user. After then asking him either he want to change any of the three, if yes then which country and either all information of that particular country or any single attribute, for which if else statements are used to make decision in between above statement.

3 Conclusion

We discussed and implemented the classes, constructor and destructor deeply with all of its variations. Also learned about access specifiers which allow the access of the members of class to outside in short they define their scope. The object concept of a class, how through an object of a class we can access each member and how to define multiple objects in just one line of code.