# Object Oriented & Programming Language

# Lab#03

## Table of Contents

## Classes

A class is basically a structure with member functions as well as member data. Classes are central to the programming methodology known as object-oriented programming.

## Defining Classes and Member Functions

A **class** is a type that is similar to a structure type, but a class type normally has member functions as well as member variables

## Member Function Definition

A member function is defined similar to any other function except that the class name and the scope resolution operator, ::, are given in the function heading.

### *Syntax*

Note that the member variable (month and day) are not preceded by and object name and do not when they occur in member function definition.

In the function definition for a member function, you can use the names of all members of the class (both the data members and the function members) without using the dot operator.

## The Dot Operator and the Scope Resolution Operator

Both the dot operator and the scope resolution operator used with member names to specify of what thing they are member.

## A Class is a Full-Fledged Type

A class is a type just like **int** and double. You can have variables of a class type, you can have parameter of class type, a function can return a value of a class type, and more generally, you can use a class type like any other type.

### Public & Private Members

There is not universal agreement about whether the public members should be listed first or the private members should be listed first. The majority seem to prefer listing the public members first. This allows for easy viewing of the portions programmers using the class actually get to use. You can make your own decision on what you wish to place first, but the examples in the book will go along with the majority and list the public members first, before the private members.

In one sense, C++ seems to favor private member first. If the first group of members has neither the public: nor the private: specifier, then members of that group will automatically be private. You will see this default behavior use in code and should Familiar with it. However, we will not use it in this book.

## Classes' Implementation

A class is a combination of data and functions joined together to form an object type. The object type defines the operations that are necessary to manipulate the object. By creating a class, we can take advantage of data hiding, which is very important in programming. Data hiding is physically located data in the class (inside the class object type) that is hidden from the "outside", which is everything in the program not defined in the class type. The data can only be accessed or changed by functions that are apart of the class. These member functions that provide access to the hidden data are called accessor methods. The declaration of a class is similar to that of a structure, except that usually there will be member functions along with data members. There will also be some type of syntax for distinguishing members that are available for public use and ones that will be kept private.

The best way to learn classes is by studying a simple class object type. The following is a very simple class object type:

```
class myClass

{

     private :

                    int data1;

     public :

                    void setData( int d )

                        {    data1 = d;     }

                    void displayData( )

                        {    cout << endl << "Data is " << data1;     }

};
```

```
#include <iostream>
using namespace std;
===================== class example  ========================

class CRectangle {
private:
int x, y;

public:
void set_values (int,int);
int area (void)
{
        return (x*y);
}
};

void CRectangle::set_values (int a, int b)

{
x = a;
y = b;

}

int main ()
{
CRectangle rect, rectb;
rect.set_values (3,4);
rectb.set_values (5,6);

cout << "rect area: " << rect.area() << endl; cout << "rectb area: " << rectb.area() << endl;
system("PAUSE");
}
/*
rect area: 12
rectb area: 30

Press any key to continue . . .

*/
```