

Lab 06
Classes

OBJECTIVE:

Things that will be covered in today's lab:

- Classes
- C++ Structs

THEORY:

Classes are an expanded concept of *data structures*: like data structures, they can contain data members, but *classes* can also contain functions as members.

Classes have the same format as plain *structures*, except that they can also include functions and have these new things called *access specifiers*. An *access specifier* is one of the following three keywords: **private**, **public** or **protected**. These specifiers modify the access rights for the members that follow them:

- Private members of a class are accessible only from within other members of the same class (or from their "*friends*").
- Protected members are accessible from other members of the same class (or from their "*friends*"), but also from members of their derived classes.
- Finally, 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 any other *access specifier* has private access automatically.

Class Definition: Classes are defined using either keyword *class*, with the following syntax:

```
class class_name
{
  access_specifier:
  //data member;
                               //member functions;
  access_specifier:
  //data member;
                               //member functions;
};
```

Access Class Members:

To access any member variable or a function of the class, we use the *member access operator* (.). The member access operator is coded as a period between the class object and member variable or function of class.

Define class member function:

In outside definition, the operator of scope (::) is used to specify that the function being defined is a member of the class and not a regular non-member function.

```
return_type class_name:: function_name( arguments )
```

C++ *structs* are almost similar to classes and have the same syntax for definition. The only difference is that the reserve word *struct* is used and all functions and data member within a class are *pubic* by default instead of *private* as in classes.

Exercise 1:

Copy and paste the following code in a text editor and check output. What is the difference between C++ Structs and Classes?

<pre>struct facultyMember { private: int id; public: void rec(int value); void printdetails(); }; int main() { facultyMember f; f.rec(10); f.printdetails(); return 0; } void facultyMember::rec(int val) { id=val; } void facultyMember::printdetails() { cout<<"ID : "<<id<<endl; }</pre>	<pre>class facultyMember { private: int id; public: void rec(int value); void printdetails(); }; int main() { facultyMember f; f.rec(10); f.printdetails(); return 0; } void facultyMember:: rec(int val) { id=val; } void facultyMember::printdetails() { cout<<"ID : "<<id<<endl; }</pre>
---	---

Exercise 2:

Design and Implement a class **dateType** that manages a calendar. The class **dateType** should store *day, month, and year* and display it in the following format (**01-May-1998**). For any object of **dateType**, your program should be able to perform the following operations on that object.

1. Set the date
2. Get the date
3. Display the date
4. Increment the day by 1
5. Increment the month by 1
6. Increment the year by 1
7. Create another object of **dateType** and compare it with the object you created earlier; determine whether they are equal or not.
8. Increment the day, month or year by certain number, suppose the date is 29-4-1976 and we add 4 days to it, now the date should be 3-5-1976.

Your class should be defined like this

```
class dateType
{
private:
    int day, month, year;
    string mon; //Month name

public:
    //define functions here with appropriate parameters and //return
    type.
}

```

Sample main function:

```
void main()
{
    dateType D;
    dateType C;
    int d,y,m;

    cout<<"Enter date"<<endl;
    cin>>d>>m>>y;
    cout<<endl;
    D.setdate(d,m,y);
    D.printdate();
    cout<<endl;
    cout<<"*****"<<endl;
    d=31,
    m=9;
    y=1995;
}

```

Lab Manual of Programming Fundamentals

```
cout<<"Set date:"<<d<<"-"<<m<<"-"<<y<<endl;
D.setdate(d,m,y);
cout<<endl;
D.printdate();
cout<<endl;
cout<<"*****"<<endl;

d=28, m=2, y=1998;
cout<<"Set date:"<<d<<"-"<<m<<"-"<<y<<endl;
D.setdate(28,2,1998);
D.printdate();
cout<<endl;
cout<<"*****"<<endl;

cout<<"AFTER INCREMENTING BY 1 DAY"<<endl;
D.incdays();
D.printdate();
cout<<endl;
cout<<"*****"<<endl;

cout<<"If "<<endl;
D.setdate(30,11,2007);
D.printdate();
cout<<endl;
cout<<"AFTER INCREMENTING BY 1 MONTH"<<endl;
D.incmmonth();
D.printdate();
cout<<endl;
cout<<"*****"<<endl;
cout<<"AFTER INCREMENTING BY 1 YEAR"<<endl;
D.incyyear();
D.printdate();
cout<<endl;
cout<<"*****"<<endl;

C.setdate(12,3,1999);
cout<<"New object created:"<<endl;
cout<<"DATE1:"<<endl;
C.printdate();
cout<<"DATE2:"<<endl;
D.printdate();
cout<<endl;
D.equal(C);

cout<<"*****"<<endl;
cout<<"If ";
D.printdate();
cout<<endl;
cout<<"Date after 9 days will be"<<endl;
D.incdays(9);
D.printdate();
}
```

Post Lab:

Create a class **Rational** for performing arithmetic with fractions. Write a program to test your class.

Use an integer variable to represent the *Private* data of the class-- the *numerator* and the *denominator*. Provide a *constructor* that enables an object of this class to be initialized when it is declared. The constructor should contain default values in case no initializes are provided and should store the fraction in reduced form. For example, the fraction 2/4 should be stored in object as 1 in the numerator and 2 in the denominator. Provide *public* member functions that perform each of the following tasks:

1. Add two **Rational** numbers. The result should be stored in reduced form.

Two rational numbers a/b and c/d can be added as follows:

$$(a/b) + (c/d) = (a*d + c*b) / (b*d)$$

2. Multiply two **Rational** numbers. The result should be stored in reduced form.

The product of two rational numbers a/b and c/d can be found as follows:

$$(a/b) * (c/d) = (a*c) / (b*d)$$

3. Divide two **Rational** numbers. The result should be stored in reduced form.

Two rational numbers a/b and c/d can be divided as follows:

$$(a/b) \div (c/d) = (a*d) / (b*c)$$

4. Print **Rational** numbers in the form a/b where a is the numerator and b is the denominator.