CL-1004
Object Oriented Programming- Lab
Spring’ 2023
BS-SE

Lab Manual 08

Problem 1:

1. Write a class Course that includes the following members:

Data Members	Properties Description
courseCode	Stores course code, like “CS 103”
courseTitle	Stores course title, “Computer Programming"
courseType	Programming fundamentals is of type CS course, Islamiat and history are Humanity courses, Mathematics is science course
creditHours	Stores credit hours of the course e.g. 1,2,3 & 4.
CourseTeacher	Stores teachers name, “Ali Khan”
section	Stores section in which student is registered. E.g. A, B, C etc

Member Functions:

a. Getters and setters for each data members.
b. Default constructor
c. Copy constructor

    Course (Course & c)

d. Parameterized constructor

    Course (string cc, string ct, int ch, string cther, char s)

2. Write a class Semester that includes the following members:

Data Members	Properties Description
semesterCode	Stores the code of semester, like “Spring 2018"
courseCount	Stores the number of courses registered between one and five courses.
courses pointer of Course Class	Course array (dynamically created using courseCount)

Member Functions:

a. Getters and setters for each data members.

b. Default constructor.

c. Copy constructor (Deep Copy)

Semester (Semester & s)

d. Parametrized constructor

Semester (string sc,int c, Course *courseArr)

After defining the classes, write the following global functions in Semester.cpp file:

1. **string mostTeachersCount(Semester sem)** : receives a semester as parameter and returns the teacher who teaches more than 1 course.

3. **void findTypesByCount(Semester sem) :** receives a semester, displays count of each type of course. Eg total type of CS course are 3, humanity courses are 1 and science course is 1.

Problem 2:

We want to create a class of Counter, the object of which holds the count of anything. Also we want to keep track of total objects of class and serial No of each object. Write a class Counter. This class has three private data members:

• count : An integer that holds a count value.
• objCount: An integer that holds the count of objects.
• serialNo: An integer that holds the serial number of objects of Counter class.

1.1 Write a default constructor that initializes each data member of the class such that count with 0, objCount that increments the count of the objects and serialNo with the correct serial no (object 1 should
have serial no 1, object2 should have serial no 2 and so on).

Counter()

1.2 Write a constructor that accepts an argument int c that is assigned to the data member count. Also initialize objCount that increments the count of the objects and serialNo with the correct serial no (object1 should have serial no 1, object2 should have serial no 2 and so on).

Counter(int c)

1.3 Write destructor of the class which adjust the count of Objects.

~Counter()

1.4 Create the getter-setter functions for the data members.

• void setCount(int c)
• int getCount()const
• int getSerialNo()const
• static int getObjCount()
• static int IncrementObjCount()

1.5 Define operator = that assigns the value of count to the left hand operand.

Counter operator = (const Counter & obj)

1.6 Define "-" unary operator that inverts the value of count data member for counter class and should allow the statements like c1 = −c1; or c2 = −c1;
Counter operator – ()

Problem 3:

Write a class Cube that holds distances or measurements expressed in feet and inches. This class has two private data members:

- height: An integer that holds the height.
- width: An integer that holds the width.
- area: An integer that holds the product of width and height.

2.1 Write a default and parameterized constructor with default parameters that initializes each data member of the class to 0.
Cube ()
Cube (int h, int w)

2.2 Generate appropriate getter-setter functions for the data members.

- **void setHeight(int f)**
- **int getHeight()const**
- **void setWidth(int i)**
- **int getWidth()const**
- **void setArea()**
- **int getArea()const**

2.3 Define an operator- that overloads the standard - math operator and allows one Cube object to be added to another.
Cube operator-(const Cube &obj)

2.4 Define an operator* function that overloads the standard * math operator and allows subtracting one Cube object from another.
Cube operator*(const Cube &obj)

2.5 Define an operator>= function that overloads the >= operator and returns true if the calling object contains a value greater than that of the parameter and returns false otherwise.
bool operator>=(const Cube &obj)

2.6 Define an operator< function that overloads the < operator and returns true if the calling object contains a value smaller than that of the parameter and returns false otherwise.
bool operator<(const Cube &obj)

2.7 Define an operator= function that overloads the = operator and assign one Cube object to another.
const Cube operator=(const Cube &obj)

Problem 4:

Design a class Complex for handling Complex numbers and include the following:

- real: a double
- imaginary: a double

The class has the following member functions.

3.1 A constructor initializing the number with default parameters.
Complex(double r, double i)

3.2 Getters and Setters of the class data members as given below

- **void setReal(double r)**
- **double getReal()const**
- **void setImaginary(double i)**
- **double getImaginary() const**

Overload the following operators in the class by identifying their return types.

3.3 Overload the binary + operator which adds two complex numbers.
Complex operator + (const Complex &obj)

3.4 Overload the binary -= operator which subtracts the complex number passed as argument from the caller object.
Complex operator -= (const Complex &obj)

3.5 Overload the binary * operator to multiply two complex numbers.
Complex operator * (const Complex &obj)

3.6 Overload binary assignemnt = operator, think about its return type.
Complex operator = (const Complex &obj)

3.7 Overload binary != operator which returns true if operands are equal and returns false otherwise.
bool operator != (const Complex &obj)

3.8 Overload unary - operator which returns true if the real and the imaginary parts are zero, otherwise return false.
bool operator-() const

Problem 5:

Write a class called Date that represents a date consisting of a year, month, and day. A Date object should have the following methods and operators:

You are given the operation and you have to overload the required operator for the objects of class.

- **Date(int year, int month, int day)** Constructs a new Date object to represent the date.
- **int getDay()** Returns the day value of this date; for example, for the date
- 2006/07/22, returns 22.
- **int getMonth()** Returns the month value of this date; for example, for the date
- 2006/07/22, returns 7.
- **int getYear()** Returns the year value of this date; for example, for the date
- 2006/07/22, returns 2006.
- **d2=d1** (Overload = operator to assign values.)
- **d1+=9** (Overload += operator which takes integer as argument . It moves this Date object forward in time by the given number of days.)
- **d2=d1-14** (Overload - operator which takes integer as argument . It moves this Date object backward in time by the given number of days)
- **d3=d1+d2** (Overload + operator which takes Date object as argument.)
- **d1-=d2** (Overload -= operator which takes Date object as argument.)
- **bool a=d1>d2** (Overload > operator which returns true or false.)
- **bool a=d1>=d2** (Overload >= operator which returns true or false.)
- **bool a=d1<d2** (Overload < operator which returns true or false.)
- **bool a=d1<=d2** (Overload <= operator which returns true or false.)
- **bool d1==d2** (Overload == operator which returns true or false.)