National University of Computer and Emerging Sciences

Laboratory Exercise

Computer Programming Lab
Spring 2019
Lab # 6

(CL 103)

Department of Computer Science

Objectives

Note: Carefully read the following instructions.

1. Make a word document with the convention “ROLLNO_SECTION_LABNO” and put all your C++ source code in it.
2. After every question paste a screenshot of your working code below the source code in the document file.
3. At the end, when you are done with your lab tasks, make your submission on Google ClassRoom.

DO DEALLOCATE WHATEVER MEMORY YOU ALLOCATE ON HEAP BEFORE YOUR PROGRAM HALTS.

Question # 1:

Create a SavingsAccount class. Use a static data member annual- InterestRate to store the annual interest rate for each of the savers. Each member of the class contains a private data member savingsBalance indicating the amount the saver currently has on deposit. Provide member function calculateMonthlyInterest that calculates the monthly interest by multiplying the balance by annualInterestRate divided by 12; this interest should be added to savingsBalance. Provide a static member function modifyInterestRate that sets the static annualInterestRate to a new value. Write a driver program to test class SavingsAccount. Instantiate two different objects of class SavingsAccount, saver1 and saver2, with balances of $2000.00 and $3000.00, respectively. Set the annualInterestRate to 3 percent. Then calculate the monthly interest and print the new balances for each of the savers. Then set the annualInterestRate to 4 percent, calculate the next month’s interest and print the new balances for each of the savers.

Question#2

Create a class HugeInteger that uses a 40-element array of digits to store integers as large as 40 digits each. Provide member functions input, output, add and subtract. For comparing HugeInteger objects, provide functions isEqualTo, isNotEqualTo, isGreaterThan, isLessThan, isGreaterThanOrEqualTo and isLessThanOrEqualTo—each of these is a “predicate” function that simply returns true if the relationship holds between the two HugeIntegers and returns false if the relationship does not hold. Also, provide a predicate function isZero. If you feel ambitious, provide member functions multiply, divide and modulus.

Question # 3:

To measure temperature you have: Kelvin, Celsius, Fahrenheit. Convert the temperature between these three types. This means that you could create abstract class CTemprerature, and use it as a base class for: CKelvin, CCelsius and CFarenhite. In order to convert those objects, you could use stand alone functions as friends.

Question # 4:

Create class CCalendarDate. That could be done if you have three classes: CDay, CMonth, CYear. After, you have created class CCalendarDate, you could create non member function that will calculate how many days is difference among two calendar dates. Use Friend function and friend classes

Question # 5:

In geometry, a point is a position in space. We can define a point in 3d-space as the set of coordinates x, y, and z. For example, the Point(2.0, 1.0, 0.0) would be the point at coordinate space x=2.0, y=1.0, and z=0.0.

In physics, a vector is a quantity that has a magnitude (length) and a direction (but no position). We can define a vector in 3d-space as an x, y, and z value representing the direction of the vector along the x, y, and z axis (the length can be derived from these). For example, the Vector(2.0, 0.0, 0.0) would be a vector representing a direction along the positive x-axis (only), with length 2.0.

A Vector can be applied to a Point to move the Point to a new position. This is done by adding the vector's direction to the point's position to yield a new position. For example, Point(2.0, 1.0, 0.0) + Vector(2.0, 0.0, 0.0) would yield the point (4.0, 1.0, 0.0).

Points and Vectors are often used in computer graphics (the point to represent vertices of shape, and vectors represent movement of the shape).

Note: Do use friend functions and friend classes as well.

Question # 6:

Create class IntegerSet for which each object can hold integers in the range 0 through 100. Represent the set internally as a vector of bool values. Element a[i] is true if integer *i* is in the set. Element a[j] is false if integer *j* is not in the set. The default constructor initializes a set to the so-called "empty set," i.e., a set for which all elements contain false. Provide member functions for the common set operations. For example, provide a unionOf- Sets member function that creates a third set that is the set-theoretic union of two existing sets (i.e., an element of the result is set to true if that element is true in either or both of the existing sets, and an element of the result is set to false if that element is false in each of the existing sets).Provide an intersectionOfSets member function which creates a third set which is the settheoretic intersection of two existing sets (i.e., an element of the result is set to false if that element is false in either or both of the existing sets, and an element of the result is set to true if that element is true in each of the existing sets). Provide an insertElement member function that places a new integer *k* into a set by setting a[k] to true. Provide a deleteElement member function that deletes integer *m* by setting a[m] to false. Provide a printSet member function that prints a set as a list of numbers separated by spaces. Print only those elements that are present in the set (i.e., their position in the vector has a value of true). Print --- for an empty set. Provide an isEqualTo member function that determines whether two sets are equal. Provide an additional constructor that receives an array of integers and the size of that array and uses the array to initialize a set object.
Now write a driver program to test your IntegerSet class. Instantiate several IntegerSet objects. Test that all your member functions work properly.