EECE 230 Introduction to Programming, Sections 3,4, and 12 Programming Assignment 9

Dec 11, 2012

- This programming assignment consists of 3 problems.
- It is due on Tues Dec 18 in the Lab
- Related material: Structures and Classes.
- *Lab structure and regulations:*
  - ⋆ The 3 hours Lab session is on Tuesdays in Lab rooms 1,2 and 5 from 2:00 pm to 5:00 pm. It consists of three parts:
    - *Occasional Solving Session (not graded but attendance mandatory)*
    - *Programming Assignment (graded)*
    - *Occasional graded weekly quiz*
  - ⋆ You are supposed to submit your own work. Cheating will not be tolerated and will be dealt with severely: zero grades on the programming assignments, disciplinary committee, Dean's warning.
  - ⋆ Lab attendance is mandatory. Violating this rule can lead to a failing grade.

Problem 1 (Complex number class)

Design a class *complexNumber* that defines complex numbers as an Abstract Data Type (ADT). Include the member functions:

- $complexNumber :: complexNumber(double, double)$ // constructor
- $complexNumber :: complexNumber()$ // default constructor
- $void\ complexNumber :: add(complexNumber\ \&)$
- $void\ complexNumber :: multiply(complexNumber\ \&)$
- $void\ complexNumber :: divide(complexNumber\ \&)$
- $double\ complexNumber :: norm()$ // the norm of $x + iy$ is $\sqrt{x^2 + y^2}$
- $void\ complexNumber :: print()$

For example you should be able to use the class as follows:

```
complexNumber  z(2,3);
cout<<''z = '';z.print();         // z = 2+3i
complexNumber  w;
cout<<''w = '';w.print();         // w = 0+0i
w = z;
cout<<''w = '';w.print();         // w = 2+3i
w.x = 1;
```

```
      cout<<''w = '';w.print();         // w = 1+3i
      z.add(w);
      cout<<''z = '';z.print();         // z = 3+6i
      z.multiply(w); // z = (3+6i)(1+3i)
      cout<<''z = '';z.print();         // z = -15 + 15 i
      cout<< w.norm();                  // 3.1623
```

Use the above program to test your class.
(*Note:* We will study later a better implementation of this class)

Problem 2 (point and rectangle classes)

a) Design a class *point* that defines a planar point as an Abstract Data Type.

Include the member functions:

- Non-default constructor which allows the user to initialize a planar point by specifying its x and y coordinates
- Default constructor
- Scale function

  $$void\ \ point :: scale(double\ a)$$

  This function is supposed to scale the coordinates of the point by a, i.e., multiply each by the real number a.
- Add function

  $$void\ \ point :: add(point\ q)$$

  If p and q are of type point. The function $p.add(q)$ should modify p in such a way that $p.x$ is assigned $p.x + q.x$, and $p.y$ is assigned $p.y + q.y$.

(*Note:* We will study later a better implementation of this class.)

b) Modify the *selectionSort* function so that, instead of taking an array of integers, it takes an array of points and sorts them in order of increasing x-coordinates.

Call it $xSelectionSortPoints$. Its prototype is thus

$$void\quad xSelectionSortPoints(point\ A[], int\ n)$$

c) Based on the class *point*, design a class *rectangle* that defines a rectangle as an Abstract Data Type.

In this problem, by a rectangle we mean a rectangle whose edges are either vertical or horizontal (i.e., we do not deal with rotated rectangles). Thus to specify a rectangle we only need to specify its lower left corner point and its upper right corner point.

Include the member functions:

- Non-default constructors which allows the user to initialize a rectangle by specifying its 2 defining corner points.
- Default constructor
- Area member function

  $$double\ \ rectangle :: area()$$
- Scale member function

  $$void\ \ rectangle :: scale(double\ a)$$

  This function is supposed to scale each of the corner points of the rectangle by a.

- Translate member function

  $$void\ \ rectangle :: translate(point\ q)$$

  This function is supposed to translate the rectangle by the vector q, i.e., add q to both corner points.

- The membership function

  $$bool\ \ rectangle :: containsPoint(point\ p)$$

  This function is supposed to check if a given point p is inside the rectangle.

- The pointSetIntersect function

  $$void\ \ rectangle :: pointSetIntersect(point\ A[], int\ n, point\ B[], int\ \&m)$$

  This function is supposed to store in the array B all the points in $A[0 \ldots n-1]$ which are inside the rectangle. It is supposed also to set m to the number of points in $A[0 \ldots n-1]$ which are inside the rectangle. Assume that memory is allocated for the array B before calling the function.

Write a program to test your classes and functions.

Problem 3 (dynamic array class).

Design a class *myDynamicArray* that defines a Dynamic Array of integers as an Abstract Data Type (ADT).

Include the member functions:

- Non-default constructor which allows the user to dynamically allocate the needed space
- Destructor which is supposed to free the allocated space if any
- Sequential search member function which takes as input an integer to search for in the array and returns the index if found and -1 otherwise
- *print* member function
- *insertionSort* member function which is supposed to sort the array using the insertion sort algorithm.
- *void myDynamicArray* :: *copy*(*myDynamicArray* &)
- *void myDynamicArray* :: *concatenate*(*myDynamicArray* & *B*)
  This function is supposed to update the content of the dynamic array by including the elements in *B*. (*Note:* here you have to copy to a temporary place, delete, allocate, fill, and finally delete the temporary array).

Write a program to test your class.

(*Note:* This is not a complete implementation. For instance, you cannot pass an instance of this class by value to a function (try it). We will fix this issue in the next PA ... copy constructor.)