CS 240

Data Structures and Algorithms

Spring 2014

1 Lab 02

In this lab you will be exploring operator overloading, as well as the various ways in which objects are created. For this lab, we will have two classes that need to be implemented: Point and Circle.

The two classes provided utilize ♯define statements in order to make a compiler time constant. These constants include P_DEBUG and C_DEBUG, where P_DEBUG enables printing of debug statements in the Point class. Similarly, C_DEBUG enables the printing of debug statements in the Circle class. **All debug statements have been provided and should not be modified. All changes to functions that contain debug statements should appear after the debug statements. This ensures that when debugging is enabled, the debug statements are always printed as the first thing the method does.** You can change the value of these ♯define statements (to enable or disable printing the debug statements) at your leisure.

You are to implement the missing pieces of each of these classes as outlined below. Remember to write your own driver code and test your classes as thoroughly as you can. You should also write your own makefile.

2 Point

You are being provided two files:

- `Point.h` – This file declares the Point class. In this case, a point is defined as it would be on a 2-Dimensional Euclidean Plane. Comments

are included in this file to indicate which pieces of this class you are being provided.

- `Point.cpp` – This file defines the methods and functions listed in class declaration. Comments are provided to guide you in writing the code for each function.

Notice that in `Point.cpp` all methods/functions that you need to implement are labeled with a comment `//TODO`.

For the Point class, the methods you need to complete are defined as follows:

- Default Constructor – This constructor will assign the default value for a Point. In this case, this value should be the Origin.
- Explicit Value Constructor – This constructor will assign the values to the x and y coordinates of the Point, where both values are parameters to the function.
- Operator- – Calculates the difference between two points. In this context, the difference between two points is the Euclidean distance between the pair. Note: You can use the distance formula, which is derived from the Pythagorean Theorem. In this case you will be utilizing the <cmath> library, which has already been included in the header.
- Operator<< – This overloads the output operator. Printing of a point should print the x and y coordinates as an ordered pair. For example, to point the Point whose x-coordinate is 3 and whose y-coordinate is 4, the resulting output should be: `(3,4)`
- Operator>> – This overloads the input operator. First this operator will prompt the user for an x-coordinate with the phrase "`Input X:`" and storing the input. Next, this operator will prompt the user for a y-coordinate with the phrase "`Input Y:` " and storing the input.

3 Circle

You are being provided two files:

- Circle.h – This file declares the Circle class. In this case, a circle is defined by a point on the Euclidean Plane that represents the center of the circle, and a double value that represents the radius of the circle. Comments are included in this file to indicate which pieces of this class you are being provided.

- Circle.cpp – This file defines the methods and functions listed in the class declaration. Comments are provided to guide you in writing the code for each function.

Notice that in `Circle.cpp` all methods/functions that you need to implement are labeled with a comment `//TODO`.

For the Circle class, the methods you need to complete are defined as follows:

- Default Constructor – This constructor will assign the default value for a Circle. In this case, this value should be the unit circle centered at the Origin.

- Explicit Value Constructor – This constructor will assign the values to the center and radius of the Circle, where both values are parameters to the function.

- Area – This computes and returns the area of the Circle on which it is called.

- Circumference – This computes and returns the circumference of the Circle on which it is called.

- Compare – Compares the location of a Point and a Circle. The Point is passed and the return value indicates where the Point is relative to the Circle. The return value should be positive if the point is inside the Circle. The return value should be zero if the point is on the curve that defines the circle. The return value should be negative if the point is outside of the Circle.

- Arclength – Returns the length of the arc that subtends the central angle whose measure (in degrees) is provided.

- Operator>> – This overloads the input operator. First this operator will prompt the user for a Point with the phrase "`Input Center Coordinates:`" and storing the input. Next, this operator will prompt the user for a radius with the phrase "`Input Radius:`" and storing the input.

- Operator<< – This overloads the output operator. Printing of a circle should print the center using the overloaded << operator for Point, and the radius of the circle. For example, a circle constructed with the default constructor will have the following output: `Center: (0,0); Radius: 1`