CSCI 132 Data Structures—Lab #1

1 Introduction

Today's lab covers:

- C++ classes and objects.
- A "Rectangle" class, writing client code and a new member function for the class.
- Compiling C++ code using the Linux command line.

Did you take CSCI131 at Holy Cross?

Yes! Then you are likely familiar with **Logos**, the math/cs department compute server. And like CSCI131, for CSCI132 this semester we will use Visual Studio Code, Linux, the command line, and the same assignment submission/auto-grading system.

No? Then you likely aren't familiar with Linux or the math/department compute server, named **logos**, and that's fine. We have prepared a tutorial to help you become familiar with Logos, Linux, and the command line. You can start the tutorial now, or come back to it later if you would rather try the lab first. Just ask for help with anything you don't recognize!

2 Logos Setup

After logging into logos, open a terminal and run the following command to setup your environment for the class: `~csci132/bin/setup`

Try the `clang++` compiler on logos for C++.

We will be using `clang++`, from the LLVM compiler collection, to compile our C++ source code. Try `clang++ -v` on logos. This command should present you with a short message about the version of the compiler.

3 Start the lab

A 'labs' folder should have already been created for you.

- Copy the code for lab1 to your 'labs' directory using the following command in the terminal:

  `"cp -r ~csci132/labs/lab1 ~/labs/"`

- Change into this new directory (using `cd ~/labs/lab1`) and make sure you have all the files: `ls -l` should show C++ files named `rectangle.h`, `rectangle.cpp`, and `main.cpp`, along with a `lab1.txt` file for your responses to lab questions.
- We will be using Visual Studio Code (vscode) this semester. This is a fairly popular "modern" code editor, with syntax coloring, support for access to remote files, and other helpful features. Start vscode from the command line (type 'vscode' or 'code') or from the sidebar/desktop icon, then open your 'lab1' folder by selecting it in vscode. You can keep your regular terminal window open alongside vscode, or you can open a fresh terminal inside the vscode bottom tab using the terminal dropdown in the menubar.
- Open `lab1.txt`, a plain text file that should already be in your lab1 folder. Put in your name and email, etc. Introduce yourself to your neighbors and put their names in your `lab1.txt` file.

- Look at all the data member variables and member method functions already implemented in the `Rectangle` class (i.e., in `rectangle.h`). Write down what they do in the `lab1.txt` file.

Important:
For each lab, you will have one or more in-lab checkpoint, and you MUST finish tasks in the checkpoints during the lab hours. You will have till the end of the grace period for finishing the rest of the lab tasks. The checkpoints will be indicated by boxes like this, so look out for them.

4 Writing Client Code

For your first piece of code, you will write some client code using already implemented public member methods of the `rectangle` class. In the `main.cpp` file, you will see code that takes input from the user for the length and width for two rectangles and then compares the two rectangles by using their area.

Checkpoint 1:
Finish the implementation of the `compareArea` function. Return 1 if the first rectangle is larger, 0 if both rectangles have equal area, and -1 if the second rectangle is larger. Compile the code and run it using the following commands:

```
clang++ main.cpp rectangle.cpp -o rectangles

./rectangles
```

5 Implementing New Member Functions

Next, you will write the implementation of a couple of member functions. Both functions have stubs written for you in the file `rectangle.cpp`.

- **`findPerimeter()`**: `findPerimeter()` is a function similar to `findArea()`, where you calculate the perimeter of a rectangle. The perimeter is the sum of all sides of a rectangle.
- **`printDetails()`**: `printDetails()` prints all the details of a rectangle. Here is an example, please follow this exact format.

```
Length and width of the rectangle: 7 6
Area of the rectangle: 42
Perimeter of the rectangle: 26
```

Checkpoint 2:
Finish implementing the `printDetails()` and `findPerimeter()` functions as stated above. Then uncomment the appropriate lines in `main.cpp` and compile and run the code.

6 Finishing the Lab

To finish the lab, you have to write more client code. Uncommenting the next part of `main.cpp` creates a `vector` *rect* of Rectangle objects and assigns random lengths and widths to them. `vector`'s are a special type of C++ array. One major benefit for this over regular C++ arrays are that you do not need to declare a fixed size for the array at the beginning and you can keep adding elements at the end using the `push_back` method.

Complete the `findLargest()` function in `main.cpp`. This function takes as input a vector of Rectangle objects and returns the Rectangle with the largest area. You can see code in `main.cpp` to get an idea on how to loop over C++ vectors.

7 What to Turn In

Use the following command to submit your files: `~csci132/bin/submit`

You can submit as many times as you like, up to the grace period, with no penalty. Some portions of the lab may be graded by hand—those portions will show zero points in the auto-grading output.