Kyle O’Connor

Smit Patel

Saylee Dharne

Evan Akers

Lab Report 3: Vectors

This lab explores concepts of constructors, constructor overloading, functions, function overloading, operator overloading, using multiple files in a program, abstraction, polymorphism and object oriented programming. These concepts are essential to CS and Engineering because there are many times that they are needed. Object oriented programming is very important, not only because it allows code to model interactions of objects in the real world, but also because it makes code easy to read, easy to reuse easy to maintain and easy to develop. Constructors and functions are fundamentals of object oriented programming and are widely used with different applications. Overloading constructors and operators is very important because they are both examples of polymorphism. Polymorphism allows one to use constructors and operators in different ways and prevent repetition of code. This makes the code shorter, more concise and, hence, more efficient. Abstraction was used to prevent the vector variables from being changed in any way other than through specified functions, reducing the complexity of the code and protecting the information of the class. Using multiple files is also very important to Computer Science because many times it is easier to divide code into multiple files to make the code easy to reuse and more readable.

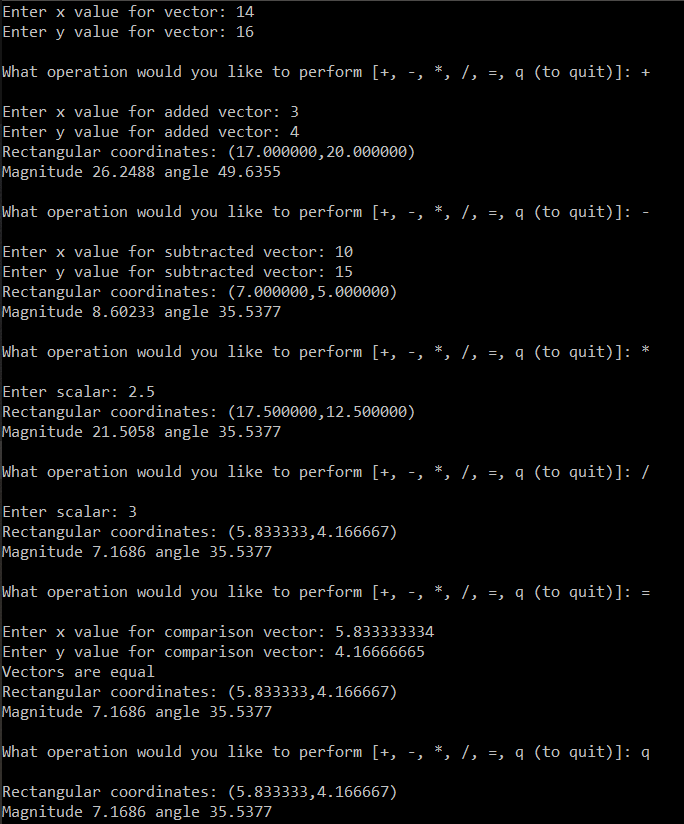
Initially, we designed the class to have 4 private member variables xVal, yVal, magnitude and angle, all of which would be set in the constructors and all of which had their own getter and setter functions. We made the xVal, yVal and magnitude to be of type double and the angle to be of type float. This allowed us to distinguish between the overloaded constructors (one accepted two double values and the other accepted one value of double type and another of float type). We decided the angle would be of type float as opposed to type double because the angle is usually approximated to 2 decimal places by convention. This seemed to work well for us for Task 1.

While working on Task 2, questions of how the magnitude and angle would change with the x and y coordinates and vice versa arose. We began to wonder whether we should include in the setter how setting the given variable would change all the other variables. We also became concerned with the sine and cosine functions of C++ which required inputs in radians as opposed to degrees, adding many angle conversions to the code. We were convinced of the need to change the structure of the class. We decided that the magnitude and angle would no longer be variables in the class. Both would be calculated as and when needed (like in one of the overloading constructors or in the functions to get the magnitude or angle). This made the code much more readable and easier to write because all of the unnecessary conversions and changes depending on the changes to 1 variable were eliminated.

During Task 3, we were made to change the class once more to add the overloaded operator functions. We added the class definition and declaration for the operators we had to overload. We then added the getMagnitude () and getAngle () functions which would calculate and return the angle and magnitude of the vector respectively. We did not feel the need to add any more changes to the class other than adding the function declarations and definitions to the class.

We did not make any changes to the class while writing the code for Task 4. The following image shows the output of a test of all the overload functions.

*Figure 1: Output of Test of All Overload Functions in Task 4*



Compilation Instructions

This has been tested by creating a new project within Visual Studios with the following options:

Win32 Console Application

Create directory for solution OFF

Empty project ON

Precompiled header OFF

SDL OFF

Then:

Add the following files to the projext:

1. Lab3Task3.cpp
2. Vector.cpp to the project
3. Vector.h to the project

Build and run.

Contribution of Team Members

Amongst the individual functions,

Each person who did their operator for task 3 or other functions made appropriate changes to the header and class file (Task 2)

Smit Patel wrote the operator overload functions for + and – and the constructors for the Vector class. (Task 1, 2, and 3)

Saylee Dharne wrote the operator overload functions for / and the getter functions. (Task 1 , 2 3)

Kyle O’Connor wrote the operator overload function for \*, setter functions, and designed the 2nd vector v2 (Task 1, 2, 3, 4)

Evan Akers wrote the operator overload function for =, the print function, and the bulk of task 4. (Task 1, 2, 3, 4)