**Computational Problem Solving 2**

**Project #1 Report**

**4 Function C++ Complex Calculator**

**Author: Joseph Rondinelli**

**Section#\_\_\_\_**



[This Photo](http://miss-wrights-wiki.wikispaces.com/Yr+8+Percentages+%26+Ratios) by Unknown Author is licensed under [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/)

**Problem Statement:**

Using C++, the programmer must design and build a program the computes 2 complex numbers in both the rectangular and polar forms. These values will need to be computed by means of addition, subtraction, multiplication, division. The user must be able to swap both values and set the results of any operation to either complex number. Lastly, every input must have error checking.

**Requirements Analysis:**

Upon entering the program, the user will be presented with a menu asking for what they would like to do. The menu is graphically laid out below, each prompt the user will enter a number or character into the prompt and based on what they enter the will arrive at some output.

Complex number input format:

<number of double> <number of double> <character>

In the menu, the user can either:

1. Enter complex numbers if (e) is entered

* Enter first complex number in polar (p) or rectangular (r)
* Prints first number in both forms
* Enter second complex number in polar (p) or rectangular (r)
* Prints first number in both forms

1. Make a Calculation if (c) is entered

* Addition if + is entered
* Computes and Prints result in both forms
* Subtraction if – is entered
* Computes and Prints result in both forms
* Multiplication if \* is entered
* Computes and Prints result in both forms
* Division if / is entered
* Computes and Prints result in both forms

1. Swap inputs if (s) is entered

* Swaps current values of both numbers and reprints them

to confirm their placement

1. Transfer answer to either complex 1 or 2 if (m) is entered

* Transfer answer to complex #1 if (1) is entered
* Copies results from previous calculation to complex #1

and prints it in both forms

* Transfer answer to complex #2 if (2) is entered
* Copies results from previous calculation to complex #1

and prints it in both forms

1. Quit if (q) or (Q) is entered

* Quits current program

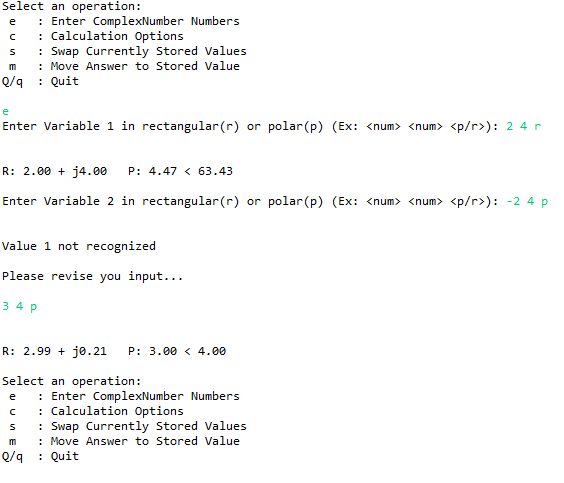
**Problems Encountered:**

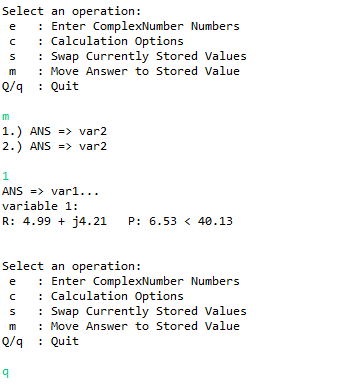
Several times during the debug errors were found that either broke the program or the IDE itself. One error that occurred was a duplication of the result of any operation being displayed in both polar and rectangular with the same values. This was due to poor handling of the data member present\_Symbol which told the converter member function if it was polar or rectangular then showData() member function would print it. At the start result has no value for present\_Symbol so converter() would fail in logic and not convert the current values contained in results object when called so depending on the calculation your result would be right for polar or rectangular but not both. Solution to this problem was setting its value before it exits converter() logic statements.

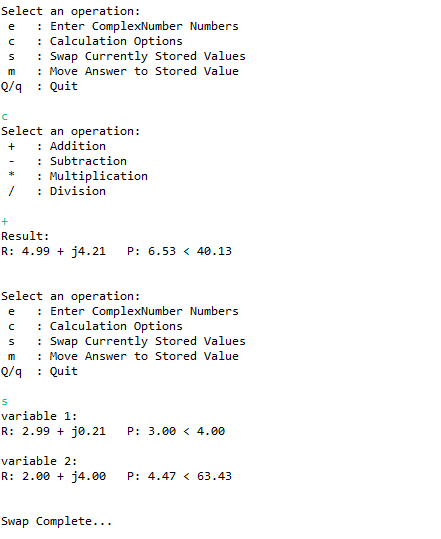
Other errors included menu formatting errors like polar and rectangular printing one then the other then switching order, simple to fix by flipping the print order. The precision of data members when printed weren’t controlled so set precision was implement along with fixed floating-point notation (fixed) to specify that I want to fix the decimal point to a certain number of digits.

**Testing:**

Testing is an important part of debugging and seeing if your final product works as intended. As a past professor told me, created test data. Pre-calculations were done to prove each operations result then the program was tested to see if the math worked out correctly. Then once the math was solid the interface was tested to see if printing and input logic was sound to see if the code would break if you enter in something wrong.

Here is an example of the code in action:





**Discussion:**

After completing this project, I realized some small issues with my approach to the plan of this project.

1. Draw diagrams and flow charts

One thing I learned in Java and VHDL type classes was the importance of drawing out the problem and figuring out how things should flow from one end of the program to another. UML is a great model to start using to show the flow of data and what types of things a class or object may need to perform. Seeing the bigger picture is essential to project success, which is why a visual representation would help you view your project.

1. Do at least enough sets of sample data

When constructing test data, you need enough tests to cover all the errors you might run into with logic. So, by creating inputs to test specific errors in logic and calculation you can quickly correct errors as they pop up.