Subhei Shaar
CS 162
2 / 6 / 2015

Assignment 2: Design/Reflection

Understanding the Problem

Create a class for rational numbers

- Numerator and denominator are the members
- Constructor to initialize the numerator and denominator as integer values
- Reducing function to put rational number in its simplest form
- negatives must be in the numerator of rational number (needs function?)

Operator Overloading

- Allow for a user to input an expression with 2 operands (the rational numbers) and a single operator
- Output the correct rational number
- Operators include addition, subtraction, division, multiplication
- Overload output operator << to print out objects from the rational number class

Inputs:

- Rational number
- Mathematical expression in the form (“ num1 / den1 operator num2 / den2 ”)

Outputs:

-Simplified rational number stored as an object

- Correctly evaluated mathematical expression using rational numbers.

Devising a Plan/Design

Class: Rational Numbers

Members: int Numerator, int Denominator

Functions:

- Rational_Numbers(int x , int y)
- Reduce() -> find GCF then divide num and den by GCF
- Convert denominator to positive if negative

Other functions:

- Command line calculator
    - Accept input in a specific form a / b operator c / d

Subhei Shaar
CS 162
2 / 6 / 2015

- Overload operators for Rational Number class
    - + - * / <<

Reflection:

This program was very easy to do because I used incremental development while coding. The first thing I decided to get working was the class of "RationalNumber". I started out with only the basic numerator and denominator as my private member variables for the class. After each implementation I would test to see if the program still compiled / generated correct output. I tested this by creating an object, setting values to the numerator and denominator, then printing this to the screen. I spent most of my time trying to develop the reduce function because I had to read up on algorithms to find the GCF. After I had done the first part of the assignment I moved on to calculating an arithmetic expression using overloaded operators. I ended up overloading 6 operators (+ - * / << >>). At first I was using getline function to get a string from the user then parsing the string by spaces. This worked but the function was very big and I tried to find a simpler way. Then I realized parsing by spaces is the same thing as using cin. This is when I thought to overload the input stream operator to make life easier. A big error I made during this assignment was writing everything in a single file to only then separate my implementation file from my interface. When I did this I got a lot of errors that I spent hours trying to resolve.

Testing

I wrote a test function that's sole purpose is to just take input of a rational number and print that object to the screen using the overloaded input and output streams. This is the way I tested the 5 different combinations of rational numbers.

- a/b where a and b have no common factors
- a/b where a and b have a common factor
- a/b where a is - and b is -
- a/b where a is - and b is +
- a/b where a is + and b is –

All but the last worked perfectly. I wanted the numerator to store the negative if the value was negative. I had to write a simple function that tested when the denominator was negative to make it positive and set the numerator to be negative. I did test 0 as the denominator and it gives a floating exception core dumped error. This is ok because I can assume the user will enter valid input for this assignment.

As for the expression calculator this took me a while to figure out. The first problem I had was I tried to use getline to get a whole string from the user then convert each piece to the appropriate variable. This was tedious and to make it work took 100 lines of code. When I overloaded the input stream operator it allowed me to take in 3 separate variables 2 rational number objects and the operator. I tested every operation in the expression and it gives the correct reduced number. You can't enter in the expression without a space in between the rational number and the operator. If no spaces are entered the whole expression will try to be saved as the rational number which will result in an error.