Programming Assignment 8 (operator overloading)

Prof. Zhang
Fall 2018

This lab practices overloading operators on a class that you defined in an earlier lab, rational class.

Requirement

Take the rational class that you design and implemented in lab7, update and extend the class to support the following operators on the class.

- Comparison operators: ==, !=, <, <=, >, >=
- Addition operators +, subtraction operator -
- Multiplication *, Division /,
- Negation - (which takes one parameter)
- Input operator >> (This replaces the input function you wrote in lab7)
- output operator << (This replaces the output function you wrote in lab7)

Testing and main() function

Your main function should run a loop that keeps reading an expression (of rational numbers and operators you implemented above) and display the value of the expression.

```
$rationalCal
Welcome! I will evaluate expressions involving rational numbers for you!
Enter the expression (e.g., 1/2-3/4) with binary operator, and press enter:
1/3+4/5
1/3+4/5 equals to 17/15
Continue to test binary operators(y/n):?y

Enter the expression (e.g., 1/2-3/4) with binary operator, and press enter:
1/2==2/4
(1/2 == 2/4) equals to true
Continue to test binary operators(y/n):?n

Enter the expression with uniary operator (e.g., -3/4): -1/2
-1/2 = -1/2
Continue to test uniary operators(y/n):?n
```

Bye!

Extra Credits:

- In the input operator, allow the user to enter the value for the rational as a fraction, e.g., 0.461
  Your function should figure out the numerator is 461, denominator is 1000.

- In the main fucntion, allow the user to choose whether to enter the test cases through keyboard, or to read them from a text file.

Requirement on Coding Style:

At this stage of CS2, please pay attention to the following requirements on all codes you write:

1. Separate your program into multiple files (modules): class header file, class implementation file, and driver file.
2. Write a simple Makefile for your program.
3. Write comments on top of each file
4. Write comments for each function, with the following information:
   - One line short description about the function: what the function is supposed to do
   - Describe each parameter
   - Describe the precondition
   - Describe the postcondition
   - Describe the return value
5. Use const modifier on pass-by-reference parameters that the function is not supposed to modify
6. Use const modifier on member functions which don't modify the invoking object

To submit:

submit2000 LAB8 rational.h
submit2000 LAB8 rational.cpp
submit2000 LAB8 main.cpp
submit2000 LAB8 Makefile