Programming Assignment 7 (ADT)

Prof. Zhang
Fall 2018

This lab practices define class, more particular, identifying the member variables, declaring and defining constructors, and other member functions.

Background reading:

Please review textbook for the basics of class, and the idea of Abstract Data Type (Section 10.2 and 10.3 in Savitch). This project is adapted from the Project 7 in Page 614 of Savitch book.

Requirement

Write a rational number class. Recall a rational number is composed of two integers with division indicated. The division is not carried out, it is only indicated, as in 1/2, 2/3, 15/32. You should represent rational numbers using two int values, numerator and denominator.

A principle of abstract data type construction is that constructors must be present to create objects with any legal values. You should provide the following three **constructors**:

1. one (default) constructor to make rational object without any argument. The constructor sets the rational object's numerator to 0, and denominator to 1.

2. one constructor with two int parameters. This constructor uses the two values to initialize the rational object's numerator and denominator respectively. If the denominator value is 0, the constructor should generate an assertion failure.

3. one constructor with one int parameter. The constructor should set the rational's numerator to the given parameter, and sets the denominator to 1.

You should also provide the following **member functions**:

- an **input** function that reads from standard input the value for the calling object. The input should be in the form of “2/3” or “27/51”.

- an output function that displays the calling object in the terminal. The output should also be in the form of “2/3”, i.e., numberator/denominator.

- One setter function that sets the numerator and denominator of the calling object.

- Two getter functions that return the numerator and denominator respectively.

You should also implement **a non-member function, Sum(),** that adds two rational objects. The function that takes two rational objects as parameters, and returns the sum of these rational numbers. Note the following formular for adding two rational numbers:

a/b + c/d = (a*d + b*c)/(b*d)

You can make the above function a friend function of class rational.

Main() and Testing Requirement

1. Basic testing

The following code segment demonstrates the usage of the above mentioned member functions, and constructors. Your main function should be some code like this.

rational a(1,2); //a is 1/2, first constructor is called to initialize a
a.output (); // display a in standard output, as 1/2

a.set (3,10); //set a's numerator to 1, denominator to 10
a.output(); //display 3/10

rational b(2); //b is 2/1, the second constructor is called to initialize b
b.output (); // display b in standard output, as 2/1

rational c; //the default constructor is called to initialize c to 0/1
c.output (); //You can see how the default construtor initializes the member variables...

c.input(); //read value for c from input. If the user enters 3/4, then
// c's numerator is set to 3, and its denominator is set to 4
c.output(); //Display c's value

//Now testing the Sum function
c = Sum (a,b); // c will be set to 1/2+2/1 = 5/2, see formular given above
c.output (); // it should display 5/2

2. Use rational class to calculate the following sum:

1/1 + 1/2 + 1/3 + 1/4 + … + 1/n

- First prompt the user to enter a positive integer, n.
- Dynamically allocate an array rational objects with the above entered size

*rational * fractions = new rational[n];*

- write a for loop that calls set function to set each of these objects to be 1/1, 1/2, 1/3, 1/4, ...

  fractions[3].set (1,4);

- use your Sum() function to calculate the sum (as a rational number) of all these n rational numbers, display the sum
- free up the memory used by the array by deleting the dynamic array

Hints

Please follow the order suggested below when working on this lab, always maintaining a compilable version of the code. Take an incremental approach. Write and test one function at a time!

1. Define the class rational first, i.e., list the member variables, and declare the member functions in the class, including the constructors. Please refer to notes and textbook for the special syntax for declaring constructors.
2. Implement the three constructors, test it in main, by declaring rational objects with no parameter, one paramter, and two paramters respectively.
3. implement output function, and test it in main. This will also help you verify/test the three constructors.
4. Implement input function, and test it in main.
5. Implement Sum function, and test in main.
6. Write the code to calculate the sum of sequences 1, 1/2, 1/3, … 1/n.

To submit:

submit2000 LAB7 rational.h
submit2000 LAB7 rational.cpp
submit2000 LAB7 main.cpp