**C/C++ RATIONAL NUMBERS LAB REPORT**

**1) Enter your name, student ID, platform (Mac or PC) and date**

Name and Student ID: Samuel Indurkar, 0888068

Class: CIS054 C/C++ Programming

Platform (Mac or PC): gcc and eclipse on MAC  
Date: 7/16/2017

**DESCRIPTION:**

1) Make sure that you can get the Sample Code that is provided on Moodle to work. It shows how to implement the + and += operators for the Rational class.

2) Create the code to implement and test the **–** **–**= \* \*= / /= and == operators

**LAB REPORT:  
2) Determine the Inputs, Processing and Outputs before creating the program**

|  |  |  |
| --- | --- | --- |
| **INPUTS** | **PROCESSING** | **OUTPUTS** |
| pre-determined input of n1= 0/1  n2=3/4 and  n3= 2/3 | Perform rational multiplication by doing straight thru multiplication of numerator and denominator.  For Division, just flip the numerator and denominator of the second number and perform the multiplication | display the three rational numbers n1, n2 and n3 |

**3) Fill in the EXPECTED & ACTUAL RESULTS**

|  |  |  |
| --- | --- | --- |
| **TEST DATA VALUES**  Provide examples showing implementation of the **–** **–**= \* \*= / /= and == operators | **EXPECTED RESULT**  Computed values before the program is run | **ACTUAL RESULT**  Fill in the output displayed  by the program |
| pre-determined input of n1= 0/1  n2=3/4 and  n3= 2/3 | \*\*\*\* n1 = n2 \* n3; // n1 should become 1/2. Others unchanged  \*\*\*\* n1 = n2 \*= n3; // n1 and n2 should become 1/2. Others unchanged  \*\*\*\* n1 = n3 / n2; // n1 should be 8/9. Others unchanged  \*\*\*\* n1 = n3 -= n2; // n1 and n3 should be 8/9. Others unchanged | n1 = n2 \* n3;  Before 0/1 3/4 2/3  After 1/2 3/4 2/3  n1 = n2 \*= n3;  Before 0/1 3/4 2/3  After 1/2 1/2 2/3  n1 = n3 / n2;  Before 0/1 3/4 2/3  After 8/9 3/4 2/3  n1 = n3 -= n2; Before 0/1 3/4 2/3  After 8/9 3/4 8/9 |

**DISCUSSION:**

**4) Complete the DISCUSSION section. It does not need to be long, but it needs to be complete.**4a) What did you do to develop the program? ("Followed the Directions" is not a complete description)

The program already has the pre-determined input of n1= 0/1

n2=3/4 and

n3= 2/3

And the instructor Mr. McElroy had already provided addition and subtraction as example. I wrote the multiplication and division. Perform rational multiplication by doing straight thru multiplication of numerator and denominator.

For Division, just flip the numerator and denominator of the second number and perform the multiplication

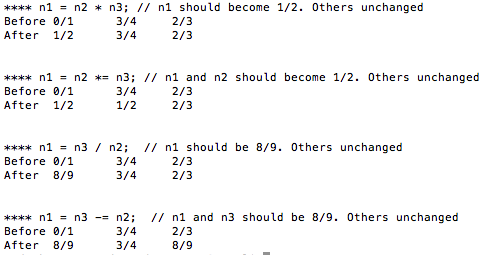
4b) What problems did you have and how did you overcome the problems?

no problem.

**PROGRAM OUTPUT:**

**5) Show one or more screen shots showing implementation of – –= \* \*= / /= and ==**

Refer to previous lab assignments for instructions on how to capture a screen or portions of a screen for either the PC or a Mac



**PROGRAM LISTING:**

**6) Copy and paste the code that YOU typed to make the program work. Your program should include a comment block at the top that shows the name of the program, date, version and your name.**

/\*

\* rational.hpp

\*

\* Created on: Jul 16, 2017

\* Author: Samuel

\*/

#ifndef RATIONAL\_H // if this compiler macro is not defined

#define RATIONAL\_H // then define it so this file will not be processed again

#include <iostream>

using namespace std;

class Rational

{

// Friend functions are actually declared outside the scope of the

// class but have the right to access public and private data and

// member function members that belong to the class. The friend

// function below gives the << operator for ostreams (including cout)

// the ability to output a Rational object by accessing its member data.

friend ostream &operator<< (ostream &out, Rational const &r);

public:

Rational (int num=0, int denom=1); // also provides default constructor

Rational add (Rational right);

Rational operator + (Rational right); // + addition operator

Rational operator += (Rational right); // += addition assignment operator

Rational operator - (Rational right);

Rational operator -= (Rational right);

Rational operator \* (Rational right); // \* multiplication operator

Rational operator \*= (Rational right); // \*= mult assignment operator

Rational operator / (Rational right); // / div operator

Rational operator /= (Rational right); // /= div assignment operator

void display();

operator double() const; // convert Rational to double

private:

int numerator;

int denominator;

// helper functions are private and not accessible by the main program

int LCD(int v1, int v2);

Rational setRational (int n, int d);

};

#endif

=================================================

Rational.cpp

.. ..

.. ..

.. ..

Rational Rational::operator \* (Rational right)

{

// create local (temporary) variables

int newNumerator;

int newDenominator;

// compute the result and save in the local variables

// the current object's numerator and denominator are not changed

newNumerator = numerator \* right.numerator ;

newDenominator = denominator \* right.denominator;

// create a new Rational object with the result and return it

return Rational(newNumerator, newDenominator);

}

Rational Rational::operator \*= (Rational right)

{

// the current object is updated with the result of the +=

numerator = numerator \* right.numerator ;

denominator = denominator \* right.denominator;

// fix the sign, reduce the fraction and return the current object

return setRational(numerator, denominator);

}

Rational Rational::operator / (Rational right)

{

// create local (temporary) variables

int newNumerator;

int newDenominator;

// compute the result and save in the local variables

// the current object's numerator and denominator are not changed

newNumerator = numerator \* right.denominator ;

newDenominator = denominator \* right.numerator;

// create a new Rational object with the result and return it

return Rational(newNumerator, newDenominator);

}

Rational Rational::operator /= (Rational right)

{

// the current object is updated with the result of the +=

numerator = numerator \* right.denominator ;

denominator = denominator \* right.numerator;

// fix the sign, reduce the fraction and return the current object

return setRational(numerator, denominator);

}

=================================================

main()

{

.. ..

.. ..

cout << endl << endl << "\*\*\*\* my new code \*\*\*\*\*\*\n\n";

cout << endl << endl << "\*\*\*\* n1 = n2 \* n3; // n1 should become 1/2. Others unchanged" << endl;

initializeNumbers (n1, n2, n3);

displayNumbers("Before", n1, n2, n3);

n1 = n2.operator\*(n3); // n2 \* n3 = 3/4 \* 2/3 = 6/12 = 1/2

displayNumbers("After ", n1, n2, n3);

cout << endl << endl << "\*\*\*\* n1 = n2 \*= n3; // n1 and n2 should become 1/2. Others unchanged" << endl;

initializeNumbers (n1, n2, n3);

displayNumbers("Before", n1, n2, n3);

n1 = n2 \*= n3;

displayNumbers("After ", n1, n2, n3);

cout << endl << endl << "\*\*\*\* n1 = n3 / n2; // n1 should be 8/9. Others unchanged" << endl;

initializeNumbers (n1, n2, n3);

displayNumbers("Before", n1, n2, n3);

//n1 = n2 - n3;

n1 = n3 / n2; // n3 / n2 = 2/3 / 3/4 = 2/3 \* 4/3 = 8/9

displayNumbers("After ", n1, n2, n3);

cout << endl << endl << "\*\*\*\* n1 = n3 -= n2; // n1 and n3 should be 8/9. Others unchanged" << endl;

initializeNumbers (n1, n2, n3);

displayNumbers("Before", n1, n2, n3);

//n1 = n2 -= n3;

n1 = n3 /= n2;

displayNumbers("After ", n1, n2, n3);

}