Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет: «Информационных технологий и прикладной математики»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 1**

**Тема: Простые классы на языке C++**

|  |  |
| --- | --- |
| Cтудент: | Королев И.М. |
| Группа: | 8О-208Б |
| Преподаватель: | Чернышов Л.Н. |
| Дата: |  |
| Оценка: |  |

Москва 2020

**Оглавление**

1. Постановка задачи………………………………………………………………...3

2. Описание программы……………………………………………………………..4

3. Набор тестов……………………………………………………………………….6

4. Результаты выполнения тестов…………………………………………………..7

5. Листинг программы……………………………………………………………...10

# 1. Постановка задачи

Создать класс Rational для работы с рациональными дробями. Рациональная (несократимая) дробь, представляется парой целых чисел (a, b), где a – числитель, b – знаменатель. Обязательно должны быть реализованы операции:

* Сложения ;
* Вычитания ;
* Умножения ;
* Деления ;
* Операции сравнения.

Должна быть реализована функция сокращения дроби reduce(), которая обязательно вызывается при выполнении арифметических операций.

# 2. Описание программы

Программа выполняет чтение данных из консоли или файла (с помощью cat | ./исполняемый файл) и выполняет заданные действия. При запуске программы появляется меню, в котором среди вариантов выбора будет либо выполнить операцию, либо завершить программу. При выборе варианта с выполнением операции, программа выводит операции, которые можно выполнить, и что они делают. Среди операций есть: сложение, вычитание, умножение, деление и операция сравнения. Ввод каждой операции выполняется по правилу: a ? b, где a, b – рациональные числа, ? – операция, которую можно выполнить.

Рациональные числа считываются из ввода как: числитель/знаменатель.

При выполнении любой из операций используется функция reduce(int numerator, int denominator), которая преобразует результат к несократимой дроби.

# Переменные и методы класса Rational

*int numerator* – числитель.

*int denominator* – знаменатель.

*void get\_rational()* – функция записи рационального числа. Считывает числитель и знаменатель.

*void print\_rational()* – функция вывода рационального числа.

*void add(Rational term1, Rational term2) –* функция сложения двух рациональных чисел.

*void sub(Rational minuend, Rational subtrahend) –* функция вычитания двух рациональных чисел.

*void mul(Rational mult1, Rational mult2) –* функция умножения двух рациональных чисел.

*void div(Rational dividend, Rational divisor) –* функция деления двух рациональных чисел.

# Описание функций

*int abs(int a) –* функция нахождения модуля числа. Нужна для дальнейшего нахождения НОД, так как остаток от отрицательного числа может быть не такой, какой должен быть. Поэтому эта функция используется, чтобы находить остаток от положительного числа.

*int nod(int a, int b)* – функция нахождения наибольшего общего делителя (НОД) для двух чисел. Функция работает по алгоритму Евклида.

*Rational reduce(int numerator, int denominator)* – функция сокращения дроби.

*void number\_comparison(Rational a, Rational b)* – функция сравнения двух чисел. В ней также присутствует алгоритм нахождения наименьшего общего кратного (НОК), чтобы привести сравниваемые два числа к общему знаменателю. Также эта функция является дружественной для класса *Rational*.

*void PrintResult(Rational a, Rational b, Rational res, char c)* – функция вывода результата, где a, b – рациональные числа, которые были введены при бинарной операции, c – символ определенной бинарной операции, res – результат операции.

*int menu()* – функция вывода диалогового меню и последующей работы с введёнными числами.

# 3. Набор тестов

Таблица 1 – Таблица тестов

|  |  |  |
| --- | --- | --- |
| test\_01.txt | test\_02.txt | test\_03.txt |
| 1 1/2 + 3/4 1 3/4 - 7/2 1 1/1 \* 3/4 1 7/6 / 5/6 1 2/3 / 9/8 1 1/2 - 5/4 1 3/2 \* 2/3 1 125/25 \* 5/6 1 43/56 | 75/76 1 13/15 | 8/9 1 1/2 | 1/2 1 5/4 | 5/4 1 23/69 - 1/3 1 56/60 \* 60/56 1 43/91 / 43/91 1 7/8 | 8/9 1 -4/6 | -5/6 1 45/90 | 2/4 1 90/180 - 180/360 1 -7/18 \* 43/15 2 | 1 4/3 + -8/7 1 4/5 - 3/2 1 128/64 + 3/1 1 -98/-98 \* 7/5 1 45/15 \* 0/0 1 225/412 + 6/98 1 334/668 - 78/91 1 -9/8 - -9/8 1 4/3 \* 9/2 1 3/4 - -8/-9 1 32/64 + 128/256 1 333/999 / 999/333 1 45/98 | 34/56 1 -12/34 | -12/35 1 97/99 | 98/99 1 34/67 | -87/56 1 23/43 | 85/171 1 56/43 /87/43 1 98/-108 / 98/108 1 12/75 \* 54/32 2 | 1 5/4 -4/5 1 78/156 + 87/194 1 43/21 + 76/32 1 1/7 \* 14/28 1 5/65 / 25/50 1 1024/4098 - 4096/16384 1 43/44 | 56/64 1 32/56 | 97/112 1 45/78 | 80/96 1 23/25 | 24/26 1 98/100 - 25/50 1 67/98 - -9/-8 1 4/5 - -3/2 1 54/65 + 81/91 1 78/105 - 43/98 1 43/67 / 90/100 1 32/87 + 55/87 1 22/45 - 98/105 1 -4/5 - 3/5 1 2/8 \* 8/16 2 |

# 4. Результаты выполнения тестов

Таблица 2 – результаты для теста test\_01.txt

|  |  |
| --- | --- |
| Ввод | Результаты |
| 1 1/2 + 3/4 | 1/2 + 3/4 = 5/4 |
| 1 3/4 - 7/2 | 3/4 - 7/2 = -11/4 |
| 1 1/1 \* 3/4 | 1/1 \* 3/4 = 3/4 |
| 1 7/6 / 5/6 | 7/6 / 5/6 = 7/5 |
| 1 2/3 / 9/8 | 2/3 / 9/8 = 16/27 |
| 1 1/2 - 5/4 | 1/2 - 5/4 = -3/4 |
| 1 3/2 \* 2/3 | 3/2 \* 2/3 = 1/1 |
| 1 125/25 \* 5/6 | 125/25 \* 5/6 = 25/6 |
| 1 43/56 | 75/76 | 43/56 < 75/76 |
| 1 13/15 | 8/9 | 13/15 < 8/9 |
| 1 1/2 | 1/2 | 1/2 = 1/2 |
| 1 5/4 | 5/4 | 5/4 = 5/4 |
| 1 23/69 - 1/3 | 23/69 - 1/3 = 0/1 |
| 1 56/60 \* 60/56 | 56/60 \* 60/56 = 1/1 |
| 1 43/91 / 43/91 | 43/91 / 43/91 = 1/1 |
| 1 7/8 | 8/9 | 7/8 < 8/9 |
| 1 -4/6 | -5/6 | -4/6 > -5/6 |
| 1 45/90 | 2/4 | 45/90 = 2/4 |
| 1 90/180 - 180/360 | 90/180 - 180/360 = 0/1 |
| 1 -7/18 \* 43/15 | -7/18 \* 43/15 = -301/270 |
| 2 | Завершение программы |

Таблица 3 – результаты для теста test\_02.txt

|  |  |
| --- | --- |
| Ввод | Результаты |
| 1 4/3 + -8/7 | 4/3 + -8/7 = 4/21 |
| 1 4/5 - 3/2 | 4/5 - 3/2 = -7/10 |
| 1 128/64 + 3/1 | 128/64 + 3/1 = 5/1 |
| 1 -98/-98 \* 7/5 | 98/98 \* 7/5 = 7/5 |
| 1 45/15 \* 0/0 | 45/15 \* 0/1 = 0/1 |
| 1 225/412 + 6/98 | 225/412 + 6/98 = 12261/20188 |
| 1 334/668 - 78/91 | 334/668 - 78/91 = -5/14 |
| 1 -9/8 - -9/8 | -9/8 - -9/8 = 0/1 |
| 1 4/3 \* 9/2 | 4/3 \* 9/2 = 6/1 |
| 1 3/4 - -8/-9 | 3/4 - 8/9 = -5/36 |
| 1 32/64 + 128/256 | 32/64 + 128/256 = 1/1 |
| 1 333/999 / 999/333 | 333/999 / 999/333 = 1/9 |
| 1 45/98 | 34/56 | 45/98 < 34/56 |
| 1 -12/34 | -12/35 | -12/34 < -12/35 |
| 1 97/99 | 98/99 | 97/99 < 98/99 |
| 1 34/67 | -87/56 | 34/67 > -87/56 |
| 1 23/43 | 85/171 | 23/43 > 85/171 |
| 1 56/43 /87/43 | 56/43 / 87/43 = 56/87 |
| 1 98/-108 / 98/108 | -98/108 / 98/108 = -1/1 |
| 1 12/75 \* 54/32 | 12/75 \* 54/32 = 27/100 |
| 2 | Завершение программы |

Таблица 4 – результаты для теста test\_03.txt

|  |  |
| --- | --- |
| Ввод | Результаты |
| 1 5/4 -4/5 | 5/4 - 4/5 = 9/20 |
| 1 78/156 + 87/194 | 78/156 + 87/194 = 92/97 |
| 1 43/21 + 76/32 | 43/21 + 76/32 = 743/168 |
| 1 1/7 \* 14/28 | 1/7 \* 14/28 = 1/14 |
| 1 5/65 / 25/50 | 5/65 / 25/50 = 2/13 |
| 1 1024/4098 - 4096/16384 | 1024/4098 - 4096/16384 = -1/8196 |
| 1 43/44 | 56/64 | 43/44 > 56/64 |
| 1 32/56 | 97/112 | 32/56 < 97/112 |
| 1 45/78 | 80/96 | 45/78 < 80/96 |
| 1 23/25 | 24/26 | 23/25 < 24/26 |
| 1 98/100 - 25/50 | 98/100 - 25/50 = 12/25 |
| 1 67/98 - -9/-8 | 67/98 - 9/8 = -173/392 |
| 1 4/5 - -3/2 | 4/5 - -3/2 = 23/10 |
| 1 54/65 + 81/91 | 54/65 + 81/91 = 783/455 |
| 1 78/105 - 43/98 | 78/105 - 43/98 = 149/490 |
| 1 43/67 / 90/100 | 43/67 / 90/100 = 430/603 |
| 1 32/87 + 55/87 | 32/87 + 55/87 = 1/1 |
| 1 22/45 - 98/105 | 22/45 - 98/105 = -4/9 |
| 1 -4/5 - 3/5 | -4/5 - 3/5 = -7/5 |
| 1 2/8 \* 8/16 | 2/8 \* 8/16 = 1/8 |
| 2 | Завершение программы |

# 5. Листинг программы

**CMakeLists.txt**

cmake\_minimum\_required (VERSION 3.2)

project (MyStudy)

add\_executable (oop\_exercise\_01 main.cpp functions.cpp methods.cpp)

set\_target\_properties(oop\_exercise\_01 PROPERTIES

CXX\_STANDART 14

CXX\_STANDART\_REQUIRED ON

)

**Rational.h**

#ifndef RATIONAL\_H

#define RATIONAL\_H

using namespace std;

class Rational

{

    friend void number\_comparison(Rational a, Rational b);

    private:

    int numerator; // Числитель

    int denominator; // Знаменатель

    public:

    Rational(): numerator(0), denominator(1) {}

    Rational(const int num\_value): numerator(num\_value), denominator(1) {}

    Rational(const int num\_value, const int denom\_value): numerator(num\_value),

 denominator(denom\_value)

    {

        if (denominator == 0)

        {

            cout << "Denominator value cannot be 0\n";

// Выводит сообщение, что значение знаменателя не может быть равным 0

            denominator = 1;

            cout << "The denominator has been changed to 1\n";

// Выводит сообщение о том, что значение знаменателя было изменено на 1

        }

        if (denom\_value < 0)

        {

            numerator = -numerator;

            denominator = -denominator;

        }

    }

    void get\_rational(); // Функция получения рационального числа

    void print\_rational(); // Функция вывода рационального числа

    void add(Rational term1, Rational term2); // Функция сложения дробей

    void sub(Rational minuend, Rational subtrahend); // Функция вычитания дробей

    void mul(Rational mult1, Rational mult2); // Функция умножения дробей

    void div(Rational dividend, Rational divisor); // Функция деления дробей

    ~Rational() {}

};

int abs(int a);

int nod(int a, int b);

Rational reduce(int num, int den);

void PrintResult(Rational a, Rational b, Rational res, char c);

int menu();

#endif

**methods.cpp**

#include <iostream>

#include "Rational.h"

using namespace std;

void Rational::get\_rational()

{

    char c;

    cin >> numerator >> c >> denominator;

    if(denominator < 0)

    {

        denominator = -denominator;

        numerator = -numerator;

    }

    if (denominator == 0)

    {

        denominator = 1;

    }

}

void Rational::print\_rational()

{

    cout << numerator << "/" << denominator;

}

void Rational::add(Rational term1, Rational term2)

{

    Rational summ;

    summ.numerator = term1.numerator \* term2.denominator + term2.numerator \*

term1.denominator;

    summ.denominator = term1.denominator\*term2.denominator;

    summ = reduce(summ.numerator, summ.denominator);

    numerator = summ.numerator;

    denominator = summ.denominator;

}

void Rational::sub(Rational minuend, Rational subtrahend)

{

    Rational residual;

    residual.numerator = minuend.numerator \* subtrahend.denominator - subtrahend.numerator \* minuend.denominator;

    residual.denominator = minuend.denominator \* subtrahend.denominator;

    residual = reduce(residual.numerator, residual.denominator);

    numerator = residual.numerator;

    denominator = residual.denominator;

}

void Rational::mul(Rational mult1, Rational mult2)

{

    Rational composition;

    composition.numerator = mult1.numerator \* mult2.numerator;

    composition.denominator = mult1.denominator \* mult2.denominator;

    composition = reduce(composition.numerator, composition.denominator);

    numerator = composition.numerator;

    denominator = composition.denominator;

}

void Rational::div(Rational dividend, Rational divisor)

{

    Rational quotient;

    quotient.numerator = dividend.numerator \* divisor.denominator;

    quotient.denominator = dividend.denominator \* divisor.numerator;

    quotient = reduce(quotient.numerator, quotient.denominator);

    numerator = quotient.numerator;

    denominator = quotient.denominator;

}

**functions.cpp**

#include <iostream>

#include "Rational.h"

using namespace std;

int abs(int a) // Функция нахождения модуля

{

    if(a < 0)

    {

        a = -a;

    }

    return(a);

}

int nod(int a, int b) // Функция нахождения НОД с помощью алгоритма Евклида

{

    a = abs(a);

    while(a!=0)

    {

        int t = a;

        a = b % a;

        b = t;

    }

    return b;

}

Rational reduce(int numerator, int denominator) // Функция сокращения дроби

{

    int den = nod(numerator, denominator);

    numerator /= den;

    denominator /= den;

    Rational red(numerator, denominator);

    return red;

}

void number\_comparison(Rational a, Rational b) // Функция сравнения дробей

{

    if(a.numerator > 0 && b.numerator<=0)

    {

        a.print\_rational();

        cout <<" > ";

        b.print\_rational();

        cout << "\n";

    }

    else if (a.numerator <= 0 && b.numerator>0)

    {

        a.print\_rational();

        cout <<" < ";

        b.print\_rational();

        cout << "\n";

    }

    else if (a.numerator == 0 && b.numerator == 0)

    {

        a.print\_rational();

        cout <<" = ";

        b.print\_rational();

        cout << "\n";

    }

    else

    {

        int a\_n = a.numerator;

        int a\_d = a.denominator;

        int b\_n = b.numerator;

        int b\_d = b.denominator;

        int n = nod(a\_d, b\_d);

        int nok = (a\_d / n) \* b\_d;

        a\_n = a\_n \* (nok / a\_d);

        b\_n = b\_n \*(nok / b\_d);

        if(a\_n > b\_n)

        {

            a.print\_rational();

            cout << " > ";

            b.print\_rational();

            cout << "\n";

        }

        else if(a\_n < b\_n)

        {

            a.print\_rational();

            cout << " < ";

            b.print\_rational();

            cout << "\n";

        }

        else

        {

            a.print\_rational();

            cout << " = ";

            b.print\_rational();

        }

    }

}

void PrintResult(Rational a, Rational b, Rational res, char c)

// Функция вывода результата после выполнения операции

{

    a.print\_rational();

    cout <<" " << c << " ";

    b.print\_rational();

    cout << " = ";

    res.print\_rational();

    cout <<"\n";

}

int menu() // Меню действий

{

    while(true)

    {

        int variant;

        cout << "Choose an action:\n";

        cout << "1 - Perform operation\n";

        cout << "2 - Exit\n";

        cin >> variant;

        if(variant == 1)

        {

            cout << "Work with rational numbers(a,b - numbers):\n";

            cout << "a + b - Addition\n";

            cout << "a - b - Subtraction\n";

            cout << "a \* b - Multiplication\n";

            cout << "a / b - Division\n";

            cout << "a | b - Comparing numbers\n";

            char c;

            Rational a,b;

            Rational result;

            a.get\_rational();

            cin >> c;

            b.get\_rational();

            switch (c)

            {

            case '+':

                {

                    result.add(a,b);

                    PrintResult(a,b,result,c);

                }

                break;

            case '-':

                {

                    result.sub(a,b);

                    PrintResult(a,b,result,c);

                }

                break;

            case '\*':

                {

                    result.mul(a,b);

                    PrintResult(a,b,result,c);

                }

                break;

            case '/':

                {

                    result.div(a,b);

                    PrintResult(a,b,result,c);

                }

                break;

            case '|':

                {

                    number\_comparison(a,b);

                }

                break;

            default:

                cout << "Expression entered incorrectly\n";

                break;

            }

            cout << "\n";

        }

        else if(variant == 2)

        {

            return 0;

        }

        else

        {

            cout << "The entered action does not exist\n";

        }

    }

}

**main.cpp**

#include <iostream>

#include "Rational.h"

using namespace std;

int main()

{

    menu();

}

Ссылка на github репозиторий: <https://github.com/HarryLiker/oop_exercise_01>