**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №2

«Дроби»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-21Б |  | преподаватель каф. ИУ5 |
| Михалёв Я.М. |  | Козлов А.Д. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2022 г.

Постановка задачи

Создать класс для представления и работы с обыкновенными дробями. Значения дробей должны быть видны на экране в виде **–2 3/4** (для отрицательных), **5/8** (для положительных, если числитель меньше знаменателя) or **3** (если знаменатель равен 1).

Сконструировать арифметические и логические операции с рациональными дробями (+, -, \*, /, >, <, ==, !=, >=, <=), а также переопределить операции потокового ввода и вывода для обыкновенных дробей. В главной программе проверить все операции для пар дробь-дробь, дробь–**int**, **int**–дробь, дробь–**double**, **double**–дробь.

Разработка алгоритма

**Класс Fraction**

Свойства:

* int top – значение числителя
* int bottom - значение знаменателя

Конструкторы:

* Fraction()
* Fraction(int a, int b = 1, int c = -12345)
* Fraction(float x, bool round = true)
* Fraction(double x, bool round = true)

Методы:

* show() – печатает дробь

Входные данные

* Два целых числа (числитель и знаменатель)
* Три целых числа (целая часть, числитель и знаменатель)
* Число типа float
* Число типа double

Перегруженные операторы

‘=’ – присваивание

‘+=’ – увеличение на

‘-=’ – уменьшение на

‘+’ - сумма двух дробей

‘-‘ – разность двух дробей

‘\*’ – произведение двух дробей

‘/’ – частное двух дробей

‘>’ – одна дробь больше другой

‘<’ – одна дробь меньше другой

‘==’ – две дроби равны

‘!=’ – две дроби не равны

‘>=’ – одна дробь больше либо равна другой

‘<=’ – одна дробь меньше либо равна другой

‘<<’ – вывод в поток

**Список функций**

**Max**

Входные данные

* Fraction f1 – первая дробь
* Fraction f2 – вторая дробь

Назначение

Возвращает большую из двух дробей

**Nod**

Входные данные

* int x – первое число
* int y – второе число

Назначение

Возвращает наибольший общий делитель чисел x и y

**Dot**

Входные данные

* float x – вещественное число
* int precision – порядок

Назначение

Возвращает количество чисел после запятой у точки

**Reverse**

Входные данные

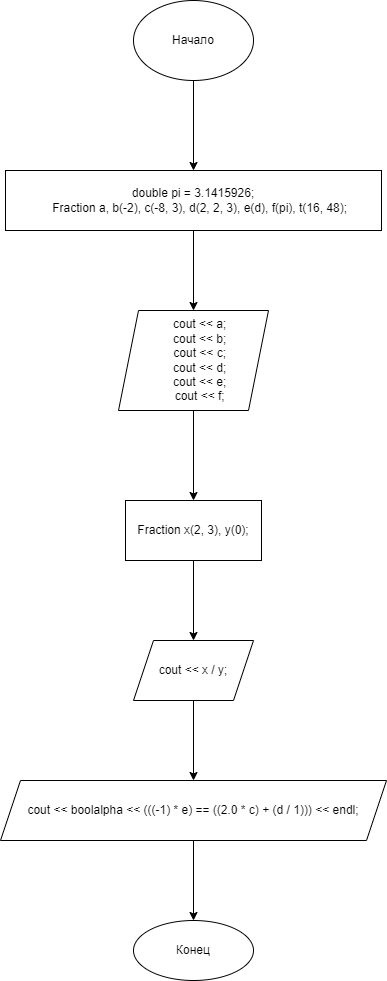
* Fraction f – дробь

Назначение

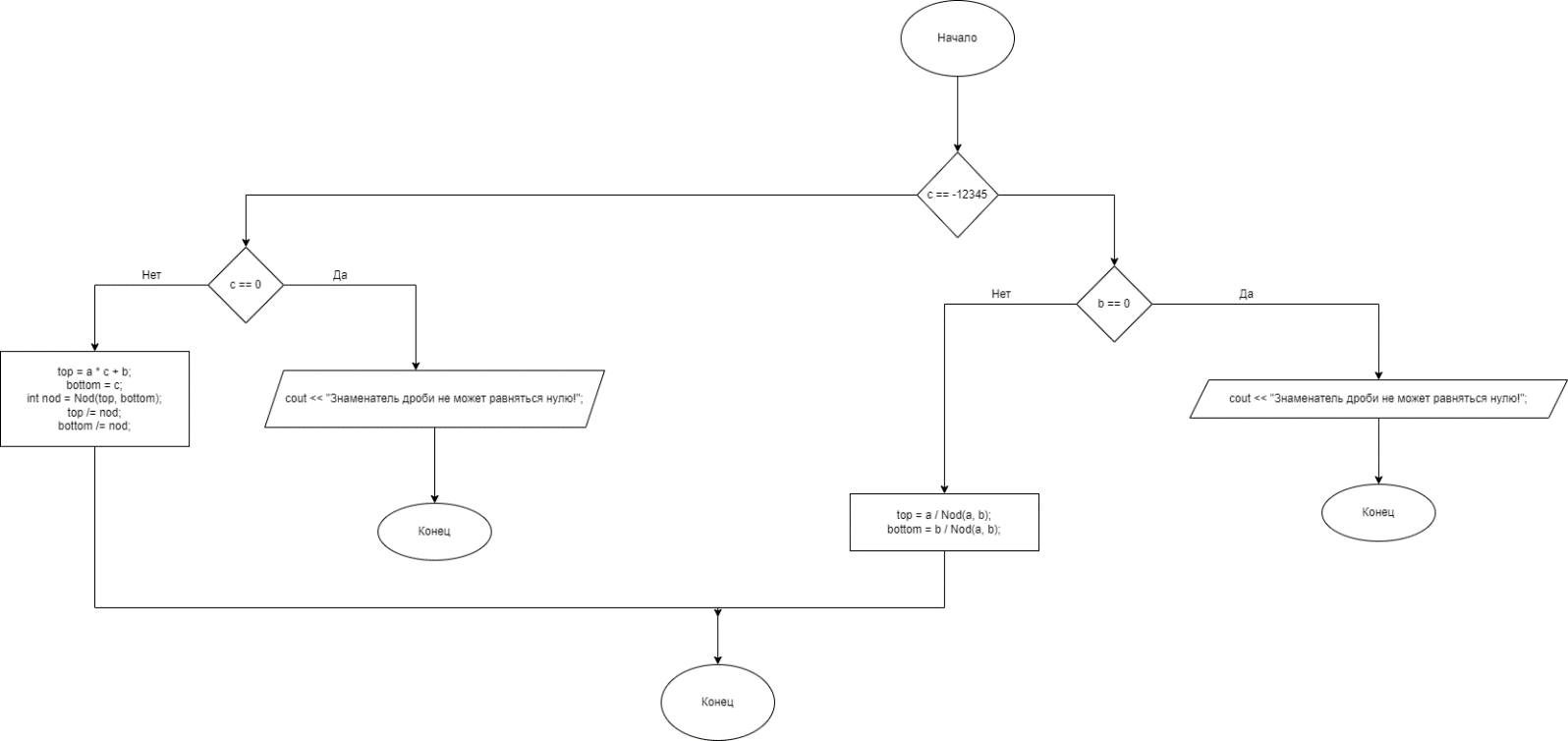
Переворачивает дробь

**Схема алгоритма**

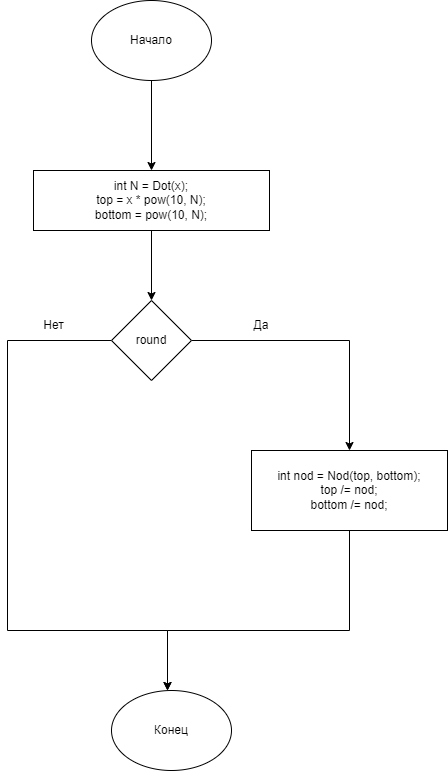
Main

****

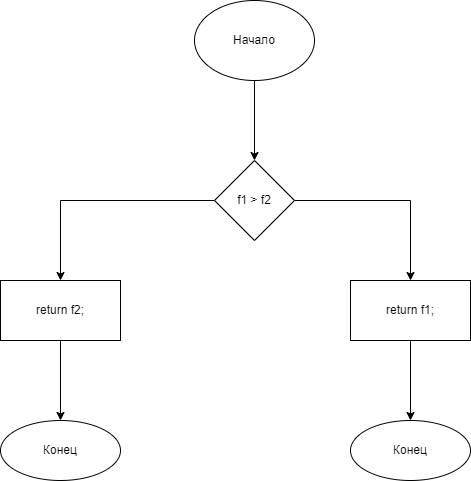
Fraction (int, int, int)



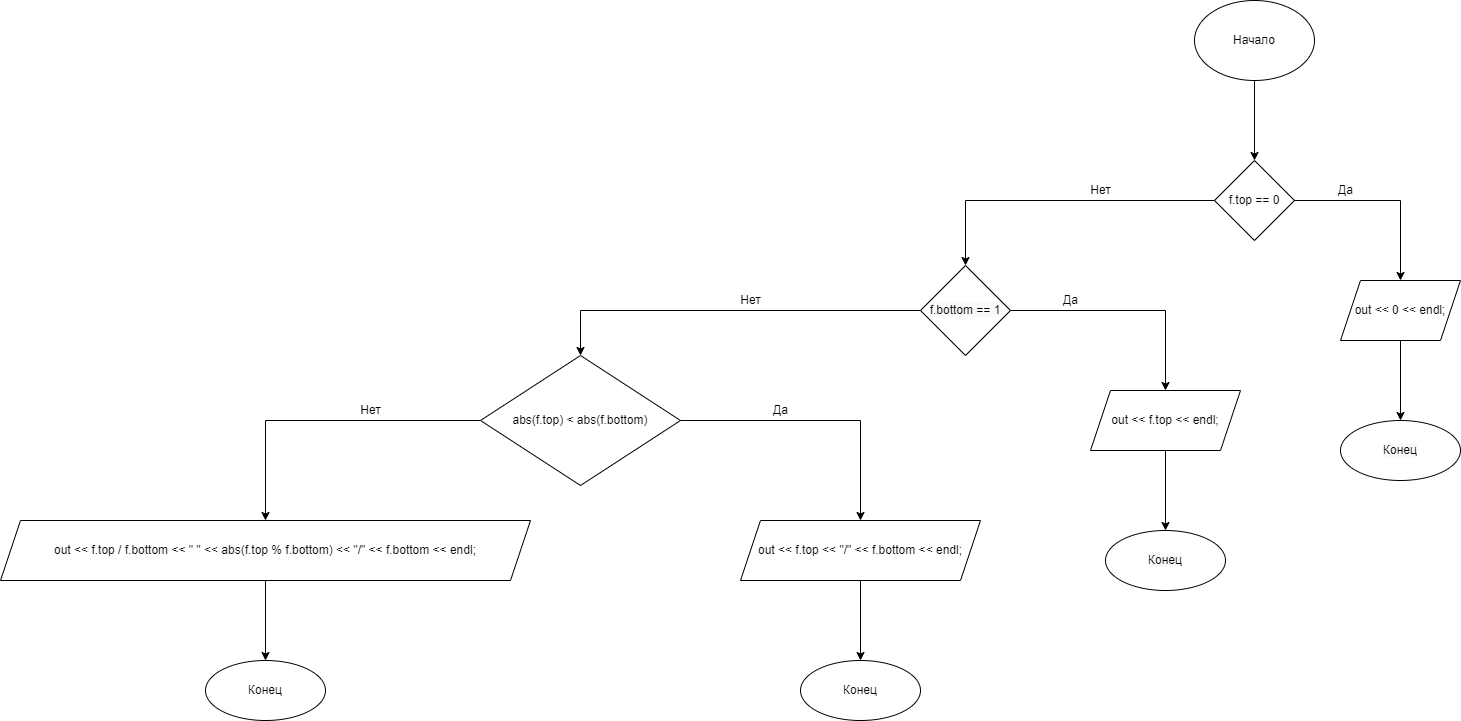
Fraction (float, round)



Max



friend void operator<<



Текст программы

Main.cpp

#include <iostream>

#include <string>

#include <format>

#include <iomanip>

#include <sstream>

#include <cmath>

#include <math.h>

#include "string.h"

#include "Utils.h"

**using** **namespace** std**;**

class Fraction

**{**

private**:**

int top**;**

int bottom**;**

public**:**

Fraction**()**

**{**

top **=** 1**;**

bottom **=** 1**;**

**}**

Fraction**(**int a**,** int b **=** 1**,** int c **=** **-**12345**)**

**{**

**if** **(**c **==** **-**12345**)**

**{**

**if** **(**b **==** 0**)**

**{**

cout **<<** "Знаменатель дроби не может равняться нулю!"**;**

exit**(**1**);**

**}**

top **=** a **/** Nod**(**a**,** b**);**

bottom **=** b **/** Nod**(**a**,** b**);**

**}**

**else**

**{**

**if** **(**c **==** 0**)**

**{**

cout **<<** "Знаменатель дроби не может равняться нулю!"**;**

exit**(**1**);**

**}**

top **=** a **\*** c **+** b**;**

bottom **=** c**;**

int nod **=** Nod**(**top**,** bottom**);**

top /= nod;

bottom /= nod;

}

}

Fraction(float x, bool round = true)

{

int N = Dot(x);

top = x \* pow(10, N);

bottom = pow(10, N);

if (round)

{

int nod = Nod(top, bottom);

top /= nod;

bottom /= nod;

}

}

Fraction(double x, bool round = true)

{

int N = Dot(x);

top = x \* pow(10, N);

bottom = pow(10, N);

if (round)

{

int nod = Nod(top, bottom);

top /= nod;

bottom /= nod;

}

}

// Конструктор копирования

Fraction(const Fraction &f)

{

top = f.top;

bottom = f.bottom;

}

// Переворачивает дробь

friend Fraction Reverse(Fraction f) { return Fraction(f.bottom, f.top); }

// Перегруженные операторы присваивания

Fraction operator= (Fraction f)

{

this->top = f.top;

this->bottom = f.bottom;

return \*this;

}

Fraction operator= (double f)

{

this->top = Fraction(f).top;

this->bottom = Fraction(f).bottom;

return \*this;

}

Fraction operator+= (Fraction f)

{

this->top += f.top;

this->bottom += f.bottom;

return \*this;

}

Fraction operator+= (double f)

{

this->top += Fraction(f).top;

this->bottom += Fraction(f).bottom;

return \*this;

}

Fraction operator\*= (Fraction f)

{

this->top \*= f.top;

this->bottom \*= f.bottom;

return \*this;

}

Fraction operator\*= (double f)

{

this->top \*= Fraction(f).top;

this->bottom \*= Fraction(f).bottom;

return \*this;

}

// Друзья класса

friend Fraction operator+ (Fraction f1, Fraction f2) { return Fraction(f1.top \* f2.bottom + f1.bottom \* f2.top, f1.bottom \* f2.bottom); };

friend Fraction operator+ (double f1, Fraction f2) { return Fraction(Fraction(f1).top \* f2.bottom + Fraction(f1).bottom \* f2.top, Fraction(f1).bottom \* f2.bottom); };

friend Fraction operator+ (Fraction f1, double f2) { return Fraction(f1.top \* Fraction(f2).bottom + f1.bottom \* Fraction(f2).top, f1.bottom \* Fraction(f2).bottom); };

friend Fraction operator- (Fraction f1, Fraction f2) { return Fraction(f1 + -1 \* f2); };

friend Fraction operator- (double f1, Fraction f2) { return Fraction(Fraction(f1) + -1 \* f2); };

friend Fraction operator- (Fraction f1, double f2) { return Fraction(f1 + -1 \* Fraction(f2)); };

friend Fraction operator\* (Fraction f1, Fraction f2) { return Fraction(f1.top \* f2.top, f1.bottom \* f2.bottom); };

friend Fraction operator\* (double f1, Fraction f2) { return Fraction(Fraction(f1).top \* f2.top, Fraction(f1).bottom \* f2.bottom); };

friend Fraction operator\* (Fraction f1, double f2) { return Fraction(f1.top \* Fraction(f2).top, f1.bottom \* Fraction(f2).bottom); };

friend Fraction operator/ (Fraction f1, Fraction f2) { return f2.top == 0 ? Fraction(1000000, 1) : f1 \* Reverse(f2); };

friend Fraction operator/ (double f1, Fraction f2) { return f2.top == 0 ? Fraction(1000000, 1) : Fraction(f1) \* Reverse(f2); };

friend Fraction operator/ (Fraction f1, double f2) { return f2 == 0 ? Fraction(1000000, 1) : f1 \* Reverse(Fraction(f2)); };

friend bool operator> (Fraction f1, Fraction f2) { return f1.top \* f2.bottom - f2.top \* f1.bottom > 0; };

friend bool operator> (double f1, Fraction f2) { return Fraction(f1).top \* f2.bottom - f2.top \* Fraction(f1).bottom > 0; };

friend bool operator> (Fraction f1, double f2) { return f1.top \* Fraction(f2).bottom - Fraction(f2).top \* f1.bottom > 0; };

friend bool operator< (Fraction f1, Fraction f2) { return f1.top \* f2.bottom - f2.top \* f1.bottom < 0; };

friend bool operator< (double f1, Fraction f2) { return Fraction(f1).top \* f2.bottom - f2.top \* Fraction(f1).bottom < 0; };

friend bool operator< (Fraction f1, double f2) { return f1.top \* Fraction(f2).bottom - Fraction(f2).top \* f1.bottom < 0; };

friend bool operator== (Fraction f1, Fraction f2) { return f1.top \* f2.bottom - f2.top \* f1.bottom == 0; };

friend bool operator== (double f1, Fraction f2) { return Fraction(f1).top \* f2.bottom - f2.top \* Fraction(f1).bottom == 0; };

friend bool operator== (Fraction f1, double f2) { return f1.top \* Fraction(f2).bottom - Fraction(f2).top \* f1.bottom == 0; };

friend bool operator!= (Fraction f1, Fraction f2) { return f1.top \* f2.bottom - f2.top \* f1.bottom != 0; };

friend bool operator!= (double f1, Fraction f2) { return Fraction(f1).top \* f2.bottom - f2.top \* Fraction(f1).bottom != 0; };

friend bool operator!= (Fraction f1, double f2) { return f1.top \* Fraction(f2).bottom - Fraction(f2).top \* f1.bottom != 0; };

friend bool operator>= (Fraction f1, Fraction f2) { return (f1.top / f1.bottom) >= (f2.top / f2.bottom); };

friend bool operator>= (double f1, Fraction f2) { return f1 >= (f2.top / f2.bottom); };

friend bool operator>= (Fraction f1, double f2) { return (f1.top / f1.bottom) >= f2; };

friend bool operator<= (Fraction f1, Fraction f2) { return (f1.top / f1.bottom) <= (f2.top / f2.bottom); };

friend bool operator<= (double f1, Fraction f2) { return f1 <= (f2.top / f2.bottom); };

friend bool operator<= (Fraction f1, double f2) { return (f1.top / f1.bottom) <= f2; };

friend void operator<< (ostream& out, Fraction f) {

if (f.top == 0)

out << 0 << endl;

else if (f.bottom == 1)

out << f.top << endl;

else

{

if (abs(f.top) < abs(f.bottom))

{

out << f.top << "/" << f.bottom << endl;

}

else

{

out << f.top / f.bottom << " " << abs(f.top % f.bottom) << "/" << f.bottom << endl;

}

}

};

};

Fraction Max(Fraction f1, Fraction f2)

{

return f1 > f2 ? f1 : f2;

}

int main()

{

system("chcp 1251 > nul");

setlocale(LC\_ALL, "rus");

double pi = 3.1415926;

Fraction a, b(-2), c(-8, 3), d(2, 2, 3), e(d), f(pi), t(16, 48);

cout << a;

cout << b;

cout << c;

cout << d;

cout << e;

cout << f;

Fraction x(2, 3), y(0);

cout << x / y;

cout << boolalpha << (((-1) \* e) == ((2.0 \* c) + (d / 1))) << endl;

return 0;

}

Utils.cpp

#include "Utils.h"

#include <string>

#include "string.h"

#include <sstream>

#include <iomanip>

**using** **namespace** std**;**

// Возвращает НОД двух чисел

int Nod**(**int x**,** int y**)**

**{**

**if** **(**y **==** 0**)** **return** x**;**

**return** Nod**(**y**,** x **%** y**);**

**}**

// Возвращает количество чисел после запятой числа x

int Dot**(**double x**,** int precision**)**

**{**

stringstream ss**;**

ss **<<** setprecision**(**precision**)** **<<** x**;**

string strNum **=** ss**.**str**();**

size\_t pos **=** strNum**.**find**(**'.'**);**

int N**;** // N - число знаков после запятой

**if** **(**pos **!=** strNum**.**npos**)**

**{**

// Число x - вещественное

N **=** strNum**.**size**()** **-** 1 **-** pos**;**

**}**

**else**

**{**

// Число x - целое

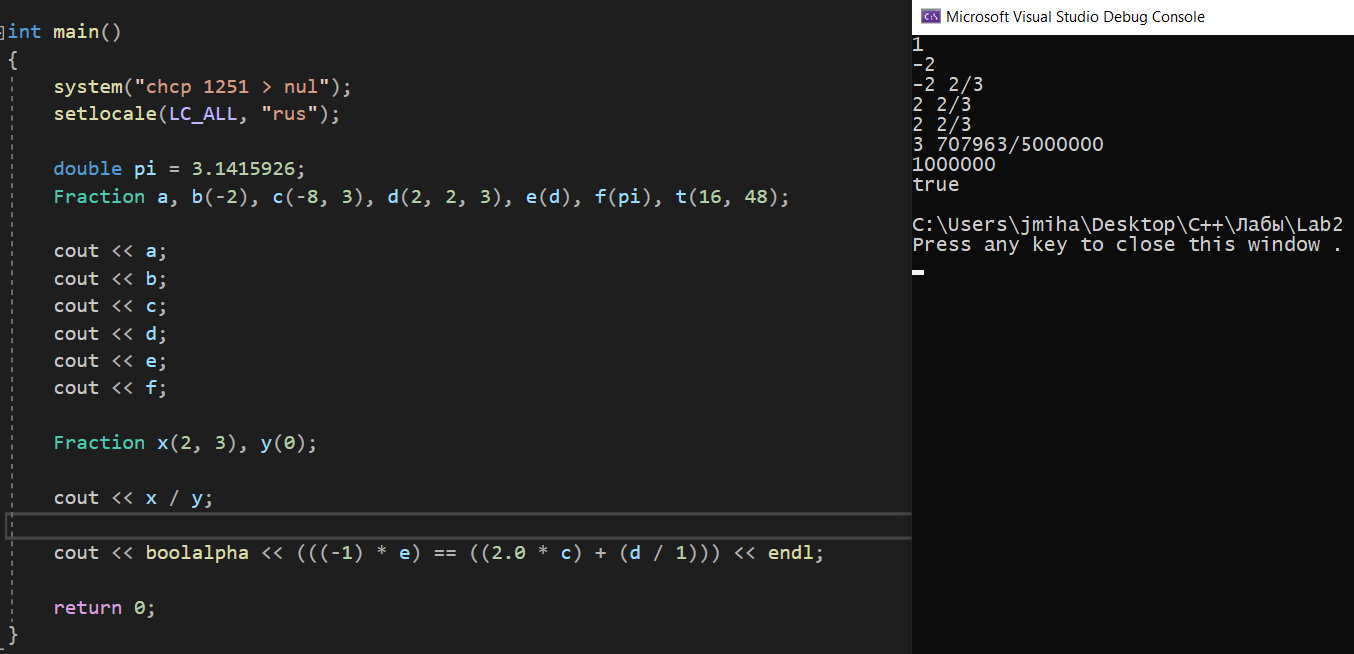
N **=** 0**;**

**}**

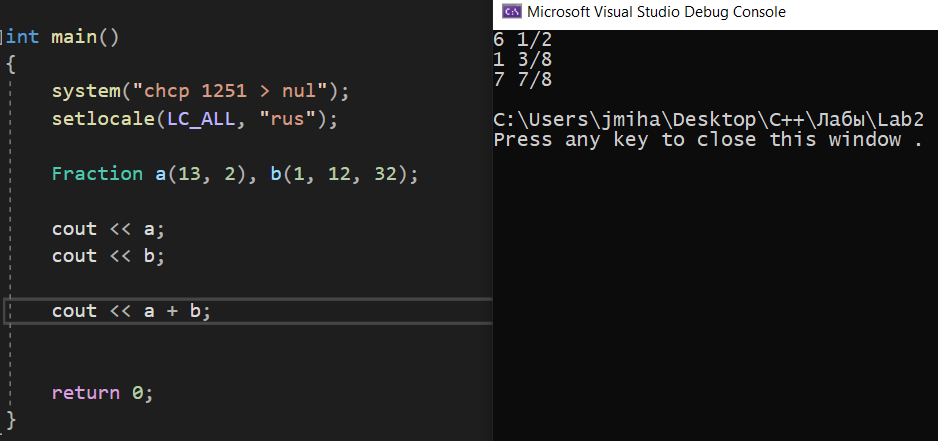
**return** N**;**

**}**

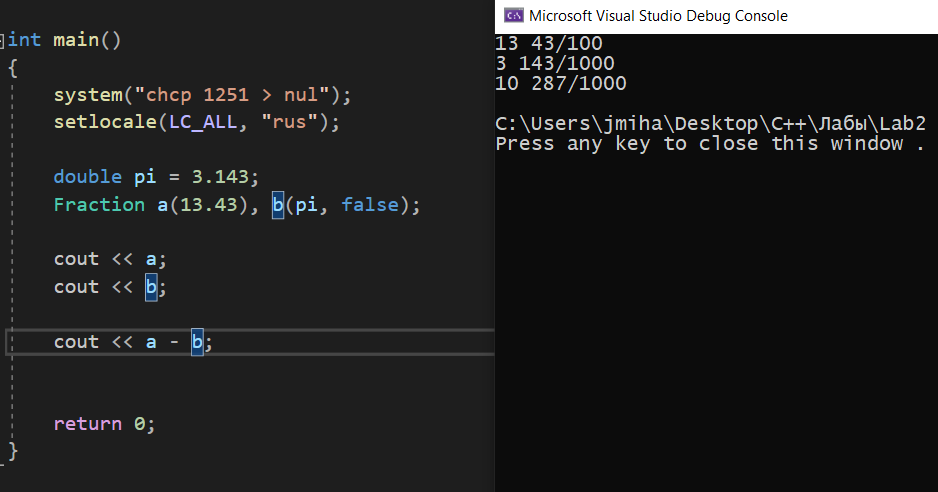
Анализ результатов



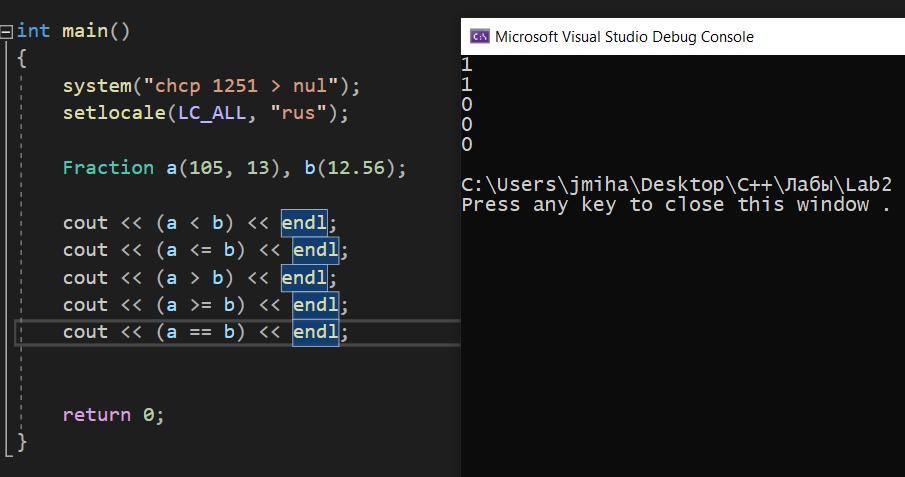
Сумма двух дробей



Инициализация дроби через вещественное число



Сравнение дробей



Вывод

Я научился

* Перезагружать операторы
* Обрабатывать дроби