**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Перегрузка операторов в С++

Студент: Тимофеев Алексей Владимирович

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи.

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

* сложения add, (*a*, *b*) + (*c*, *d*) = (*ad* + *bc*, *bd*);
* вычитания sub, (*a*, *b*) – (*c*, d) = (*ad* – *bc*, *bd*);
* умножения mul, (*a*, *b*) ´ (*c*, *d*) = (*ac*, *bd*);
* деления div, (*a*, *b*) / (*c*, *d*) = (*ad*, *bc*);
* операции сравнения.

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

Операции сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов.

1. Описание программы

Вводим в программу две рациональных дроби. Чтобы избежать непредусмотренных изменений поля класса сделаем const, поэтому инициализируем поля класса через конструктор. Все операции реализованны с дробями через перегрузку операций, также реализована операция сокращения дробей getReduced. Вывод операций сравнения реализован в bool типе, то есть если операция сравнения осуществляется правильно, выводится 1, иначе 0.

1. Набор тестов

1/2 и 1/3

5/7 и 7/9

11/13 и 11/15

6/8 и 7/8

5/6 и 5/6

1. Результаты выполнения тестов

Введите первое рациональное число :

1

--

2

Введите второе рациональное число :

1

--

3

сложение 5/6

разность 1/6

умножение 1/6

деление 3/2

больше 1

меньше 0

равно 0

Введите первое рациональное число :

5

--

7

Введите второе рациональное число :

7

--

9

сложение 94/63

разность -4/63

умножение 5/9

деление 45/49

больше 0

меньше 1

равно 0

Введите первое рациональное число :

11

--

13

Введите второе рациональное число :

11

--

15

сложение 308/195

разность 22/195

умножение 121/195

деление 15/13

больше 1

меньше 0

равно 0

Введите первое рациональное число :

6

--

8

Введите второе рациональное число :

7

--

8

сложение 13/8

разность -1/8

умножение 21/32

деление 6/7

больше 0

меньше 1

равно 0

Введите первое рациональное число :

5

--

6

Введите второе рациональное число :

5

--

6

сложение 5/3

разность 0/1

умножение 25/36

деление 1/1

больше 0

меньше 0

равно 1

1. Листинг программы

#include <iostream>

#include <cmath> // Тимофеев Алексей М8О-207Б-19

#include <cstring>

class Fraction {

private:

const long int numerator;

const long int denominator;

friend std::istream &operator>>(std::istream &in, const Fraction &fraction);

public:

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

Fraction(const long int &number) : numerator(number), denominator(1) {}

Fraction(const long int &numerator, const long int &denominator)

: numerator(numerator), denominator(denominator) {}

const long int &getNumerator() const { return numerator; }

const long int &getDenominator() const { return denominator; }

Fraction getReduced() const {

long int gcd = getGCD(getNumerator(), getDenominator());

long int numerator = getNumerator() / gcd;

long int denominator = getDenominator() / gcd;

if (denominator < 0) {

numerator = -numerator;

denominator = -denominator;

}

return Fraction(numerator, denominator);

}

static long int getGCD(const long int &a, const long int &b) {

long int x = std::abs(a);

long int y = std::abs(b);

while(1) {

x = x % y;

if (x == 0) {

return y;

}

y = y % x;

if (y == 0) {

return x;

}

}

}

};

Fraction operator+(const Fraction &a, const Fraction &b) {

return Fraction(a.getNumerator() \* b.getDenominator() +

b.getNumerator() \* a.getDenominator(),

a.getDenominator() \* b.getDenominator());

}

Fraction operator-(const Fraction &a, const Fraction &b) {

return Fraction(a.getNumerator() \* b.getDenominator() -

b.getNumerator() \* a.getDenominator(),

a.getDenominator() \* b.getDenominator());

}

Fraction operator-(const Fraction &f) {

return Fraction(-f.getNumerator(), f.getDenominator());

}

Fraction operator\*(const Fraction &a, const Fraction &b) {

return Fraction(a.getNumerator() \* b.getNumerator(),

a.getDenominator() \* b.getDenominator());

}

Fraction operator/(const Fraction &a, const Fraction &b) {

return Fraction(a.getNumerator() \* b.getDenominator(),

a.getDenominator() \* b.getNumerator());

}

std::ostream &operator<<(std::ostream &stream, const Fraction &fraction) {

Fraction reduced = fraction.getReduced();

return stream << reduced.getNumerator() << "/" << reduced.getDenominator();

}

long compare(const Fraction &a, const Fraction &b) {

return a.getNumerator() \* b.getDenominator() -

b.getNumerator() \* a.getDenominator();

}

bool operator==(const Fraction &a, const Fraction &b) {

return compare(a, b) == 0;

}

bool operator!=(const Fraction &a, const Fraction &b) {

return compare(a, b) != 0;

}

bool operator>(const Fraction &a, const Fraction &b) {

return compare(a, b) > 0;

}

bool operator<(const Fraction &a, const Fraction &b) {

return compare(a, b) < 0;

}

bool operator>=(const Fraction &a, const Fraction &b) {

return compare(a, b) >= 0;

}

bool operator<=(const Fraction &a, const Fraction &b) {

return compare(a, b) <= 0;

}

std::istream &operator>>(std::istream &in, const Fraction &fraction) {

long int a,b;

in >> a >> b;

Fraction(a,b);

return in;

}

int main() {

long int num1, num2, den1, den2;

std::cout << "Введите первое рациональное число : \n";

std::cin >> num1;

std::cout << "--\n";

std::cin >> den1;

std::cout << "Введите второе рациональное число : \n";

std::cin >> num2;

std::cout << "--\n";

std::cin >> den2;

std::cout << "сложение \t" << (Fraction(num1,den1) + Fraction(num2,den2)) << std::endl;

std::cout << "разность \t" << (Fraction(num1,den1) - Fraction(num2,den2)) << std::endl;

std::cout << "умножение \t" << (Fraction(num1,den1) \* Fraction(num2,den2)) << std::endl;

std::cout << "деление \t" << (Fraction(num1,den1) / Fraction(num2,den2)) << std::endl;

std::cout << "больше \t" << (Fraction(num1,den1) > Fraction(num2,den2)) << std::endl;

std::cout << "меньше \t" << (Fraction(num1,den1) < Fraction(num2,den2)) << std::endl;

std::cout << "равно \t" << (Fraction(num1,den1) == Fraction(num2,den2)) << std::endl;

return 0;

}

1. **Выводы**

Я научился создавать классы, перегружать операторы, и реализовать алгоритм для нахождения НОД, так же познакомился с библиотекой #include <cstring>.

1. Списоклитературы
2. ООП вкартинках/ Хабр —Habr

URL: https://habr.com/ru/post/463125/ (дата обращения: 25.09.2020)

2. Классы в C++ — урок 10/ code-live.ru

1. URL: https://code-live.ru/post/cpp-classes/#\_2