Правительство Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Отчёт к лабораторной работе №8

по дисциплине

«Языки программирования»

Работу выполнил

Студент группы СКБ222 _____ М. Д. Сосин
подпись, дата

Работу проверил _____ С. А. Булгаков
подпись, дата

Москва 2022

СОДЕРЖАНИЕ

ПОСТАНОВКА ЗАДАЧИ

Разработать класс `BigFraction`, являющийся наследником класса `fraction`
из лабораторной работы №7, использующий для хранения значений числителя
и знаменателя класс `bigint` из лабораторной работы №6.

1 Идея решения задачи

Написана программа, в которой реализован метод позволяющий описывать дроби, а также выполнять арифметические операции вида f@f, `fraction f`: умножение/деление на целое число, приведение к приближенному значению типа `double`, а также операции помещения (извлечения) в поток (из потока).

2 Структура программы

2.1 Функция main
Данная функция демонстрирует применение методов класса.

2.2 Конструктор BigFraction()
Создает пустой объект класса «BigFraction». Числитель равен 0, знаменатель – 1.

2.3 Конструктор BigFraction(const BigFraction&)

Копирует все параметры переданного объекта.

2.4 Метод ~BigFraction()
Деструктор класса.

2.5 Конструктор BigFraction(const bigint&, const bigint&)
Принимает на вход 2 числа типа bigint, сокращает дробь, создает объект класса «BigFraction». Присутствуют проверки на нулевой знаменатель.

2.6 Метод print()
Выводит числитель и знаменатель в формате: числитель/знаменатель.

2.7 Оператор умножения
Принимает на вход 2 объекта класса «BigFraction». Создает и возвращает новый объект класса «BigFraction», чей числитель равен произведению числителей, знаменатель – произведению знаменателей объектов.

2.8 Оператор сложения
Принимает на вход 2 объекта класса «BigFraction». Создает и возвращает новый объект класса «BigFraction», чей числитель равен сумме произведений числителя объекта на знаменатель другого объекта и знаменателя объекта на числитель другого объекта, знаменатель – произведению знаменателей.

2.9 Оператор деления
Принимает на вход 2 объекта класса «BigFraction». Создает и возвращает новый объект класса «BigFraction», чей числитель равен произведению числителя на знаменатель, знаменатель – произведению знаменателя на числитель.

2.10 Оператор вычитания
Принимает на 2 вход объект класса «BigFraction». Создает и возвращает новый объект класса «BigFraction», чей числитель равен разности произведений числителя объекта на знаменатель другого объекта и знаменателя объекта на числитель другого объекта, знаменатель – произведению знаменателей. Выполняется проверка на положительность результата.

2.11 Оператор умножения на число
Принимает на вход число типа unsigned long int и объект класса «BigFraction». Создает и возвращает новый объект класса «BigFraction», чей числитель равен произведению числителя на переданное число, знаменатель – знаменателю объекта.

2.11 Оператор деления на число
Принимает на вход число типа unsigned long int и объект класса «BigFraction». Создает и возвращает новый объект класса «BigFraction», чей числитель равен числителю объекта, знаменатель – произведению знаменателя объекта на переданное число.

2.12 Оператор присваивания
Принимает на вход объект класса «BigFraction», копирует значения.

3 Результаты тестирования

3.1 Тестирование метода Bigfraction Bigfraction::operator*(const fraction& r)
Пример кода запуска:

```
#include "bigint.h"
#include "fraction.h"
#include "BigFraction.h"
#include <iostream>

int main(){
    bigint f_ch("123");
    bigint f_zn("532");
    bigint s_ch("223");
    bigint s_zn("3123");
    unsigned long int  c = 3;
    BigFraction f1(f_ch, f_zn);
    BigFraction f2(s_ch, s_zn);
    BigFraction f3(f1 * c);

    f3.print();
    return 0;
}
```

Введем дробь и умножим ее на число 5.

1175/675

Все работает корректно.

Умножим дробь с длинными числами на 5.

445236624936170/2346126349619823698762784627 3

Все работает корректно.

3.2 Тестирование метода Bigfraction Bigfraction::operator/(const fraction& r)
Введем дробь и поделим её 5.

89047324987234/469225269923964739752556925 46

Все работает корректно.

Введем дробь и поделим её на 2.

235/1350

3.3 Тестирование метода Bigfraction Bigfraction::operator+(const fraction& r)
Прибавим к 275/675 дробь 89047324987234/2346126349619823698762784627 3

Получим:

551339692160664579903691025710 5/158363528599338099666487962342 75

Все работает корректно.

3.4 Тестирование метода Bigfraction Bigfraction::operator-(const fraction& r)
Введем дробь 235/ 675 и отнимем 121/12.

```
Error sub < 0
```

Введем дробь 235/10 и отнимем 100/10

```
1350/100
```

Все работает корректно.

3.5 Тестирование метода Bigfraction Bigfraction::operator*(unsigned long int r)
Введем дробь 235/131 и умножим на 12.

```
2820/131
```

Все работает корректно.

3.6 Тестирование метода Bigfraction Bigfraction::operator/(unsigned long int r)
Введем дробь 235/131 и разделим на 12.

```
235/1572
```

Все работает корректно.

ПРИЛОЖЕНИЕ А

А.1 Исходный код программы BigFractions.h:

```
#ifndef BIG_FRACTION_H_INCLUDED
#define BIG_FRACTION_H_INCLUDED
#include <cstddef>
#include "fraction.h"
#include "bigint.h"

class BigFraction : public fraction {
    bigint a;
    bigint b;
  public:
    BigFraction();
    BigFraction(const bigint& a_in,const bigint& b_in);
    BigFraction(const fraction& f);
    BigFraction(const BigFraction& f);
    ~BigFraction();
    void operator= (const BigFraction& right);
    void print();

    friend BigFraction operator* (const BigFraction& left, const BigFraction&
right);
    friend BigFraction operator/ (const BigFraction& left, const BigFraction&
right);
    friend BigFraction operator+ (const BigFraction& left, const BigFraction&
right);
    friend BigFraction operator- (const BigFraction& left, const BigFraction&
right);
    friend BigFraction operator* (const BigFraction& left, unsigned long int
right);
    friend BigFraction operator/ (const BigFraction& left, unsigned long int
right);
};

#endif // BIG_FRACTION_H_INCLUDED
```

А.2 Исходный код BigFractions.cpp

```
#include "bigint.h"
#include "fraction.h"
#include "BigFraction.h"
#include <cstring>
#include <stdlib.h>
#include <iostream>

using std::cerr;
using std::cout;
using std::endl;

BigFraction::BigFraction(): fraction(){
                a = bigint("0");
                b = bigint("1");
        }
```

```
BigFraction::BigFraction(const bigint& a_cur,const bigint& b_cur) {
    bigint ac = a_cur;
    bigint bc = b_cur;
    if (bc == 0) {
        cerr << "Error division by zero" << endl;
        exit(1);
    }
    for (unsigned long i = 2; i <= (ac + bc) / 2; ++i) {
        while ((ac % i == 0) && (bc % i == 0)) {
            ac = ac / i;
            bc = bc / i;
        }
    }
    a = ac;
    b = bc;
}

void BigFraction::print(){
    cout << a << "/" << b << endl;
}


BigFraction::BigFraction(const BigFraction& f){
        a = bigint(f.a);
        b = bigint(f.b);
}


BigFraction operator*(const BigFraction& left, const BigFraction& right) {
        BigFraction ans(left.a*right.a, left.b*right.b);
          return ans;
}

BigFraction operator/(const BigFraction& left, const BigFraction& right) {
    BigFraction ans(left.a*right.b, left.b*right.a);
    return ans;
}

BigFraction operator+(const BigFraction& left, const BigFraction& right) {
    BigFraction ans(left.a*right.b + left.b*right.a, left.b*right.b);
    ans.print();
    return ans;
}

BigFraction operator-(const BigFraction& left, const BigFraction& right) {
    if ((left.a*right.b - left.b*right.a) < 0) {
        cerr << "Error sub < 0" << endl;
          exit(1);
    }
    BigFraction ans(left.a*right.b - left.b*right.a, left.b*right.b);
    return ans;
}

BigFraction operator*(const BigFraction& left, unsigned long int right) {
    BigFraction ans(left.a*right, left.b);
    return ans;
}

BigFraction operator/(const BigFraction& left, unsigned long int right) {
    BigFraction ans(left.a, left.b*right);
    return ans;
}
```

```
void BigFraction::operator=(const BigFraction& right){
    a = bigint(right.a);
    b = bigint(right.b);
    a.print();
    b.print();
}

BigFraction::~BigFraction()
{}
```

A.4 Uml-диаграмма: