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

**СОДЕРЖАНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc121828338)

[1 Идея решения задачи 4](#_Toc121828339)

[2 Структура программы 4](#_Toc121828340)

[2.1 Функция main 4](#_Toc121828341)

[2.2 Конструктор BigFraction() 4](#_Toc121828342)

[2.4 Метод ~BigFraction() 4](#_Toc121828343)

[2.5 Конструктор BigFraction(const bigint&, const bigint&) 4](#_Toc121828344)

[2.6 Метод print() 4](#_Toc121828345)

[2.7 Оператор умножения 4](#_Toc121828346)

[2.8 Оператор сложения 4](#_Toc121828347)

[3 Результаты тестирования 6](#_Toc121828348)

[3.1 Тестирование метода Bigfraction Bigfraction::operator\*(const fraction& r) 6](#_Toc121828349)

[3.2 Тестирование метода Bigfraction Bigfraction::operator/(const fraction& r) 6](#_Toc121828350)

[3.3 Тестирование метода Bigfraction Bigfraction::operator+(const fraction& r) 6](#_Toc121828351)

[3.4 Тестирование метода Bigfraction Bigfraction::operator-(const fraction& r) 7](#_Toc121828352)

[3.5 Тестирование метода Bigfraction Bigfraction::operator\*(unsigned long int r) 7](#_Toc121828353)

[3.6 Тестирование метода Bigfraction Bigfraction::operator/(unsigned long int r) 7](#_Toc121828354)

[ПРИЛОЖЕНИЕ А 8](#_Toc121828355)

# 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.

*445236624936170/23461263496198236987627846273*

Все работает корректно.

### 3.2 Тестирование метода Bigfraction Bigfraction::operator/(const fraction& r)

Введем дробь и поделим её 5.

89047324987234/46922526992396473975255692546

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 89047324987234/23461263496198236987627846273

Получим:

*5513396921606645799036910257105/15836352859933809966648796234275*

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()

{}

А.4 Uml-диаграмма:

