**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 №3

«Классы. Перегрузка конструкторов и операций»

| Выполнил: |  | Проверил: |
| --- | --- | --- |
| студент группы ИУ5-24б |  | преподаватель каф. ИУ5 |
| Чернев Н.А. |  |  |
| Подпись и дата: 16.03.2024 |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

## Задание

Лабораторная работа состоит из двух частей:

1. Часть 1:
   1. Создать класс "Дроби" для выполнения арифметических операций над обыкновенными дробями. Внутреннее представление дробей (состав полей класса) должно обеспечивать эффективное выполнение операций над дробями и может отличаться от представления дробей на экране монитора, которое должно быть удобным для пользователя. Например, внутри объекта класса "Дроби" может храниться неправильная дробь.
   2. Разработать конструктор по умолчанию и конструктор, который преобразует строку, содержащую внешнее представление дроби, в объект класса "Дроби".
   3. Перегрузить операции потокового ввода дроби с клавиатуры и вывода ее на экран монитора. При вводе выполнять сокращение дроби.
   4. Создать многофайловый проект и отладить программу, которая создает один объект класса "Дроби" и выводит значения его полей на экран (эта программа состоит из 5 первых операторов программы, приведённой в [Приложении 1](https://iu5edu.ru/wiki/cpp2/docs/labs/lab3/Instructions/SupplementMaterial1/)).  
      ПРЕДУПРЕЖДЕНИЕ  
      Класс "Дроби" необходимо упаковать в самостоятельную статическую библиотеку.
2. Часть 2:
   1. Дополнить класс функциями-членами класса и функциями-друзьями класса, которые необходимы для выполнения программы из [Приложения 1](https://iu5edu.ru/wiki/cpp2/docs/labs/lab3/Instructions/SupplementMaterial1/).
   2. Выполнить программу из [Приложения 1](https://iu5edu.ru/wiki/cpp2/docs/labs/lab3/Instructions/SupplementMaterial1/) и сравнить результаты с тестовым примером.

**Разработка алгоритма**

Библиотека fraction:

* int gcd(int a, int b); - принимает 2 целых числа, возвращает их НОД
* Класс Fraction:
  + Поля:
    - int numerator; - числитель дроби
    - int denumerator; - знаменатель дроби
  + Методы:
    - void simplify(); - упрощает дробь
    - Fraction(); - конструктор без параметров
    - Fraction(int \_numerator, int \_denumerator); - конструктор с параметрами “числитель” и “знаменатель”
    - Fraction(const char \*str); - конструктор с параметром char \*
    - Fraction(const Fraction &fr); - конструктор копирования
    - Fraction(double dbl); - конструктор с параметром double
    - int getNumerator(); - возвращает числитель
    - void setNumerator(int \_numerator); - записывает в числитель передаваемое значение
    - int getDenumerator(); - возвращает знаменатель
    - void setDenumerator(int \_denumerator); - записывает в знаменатель передаваемое значение
    - Fraction operator + (Fraction fr); - перегрузка оператора + для Fraction + Fraction
    - Fraction operator + (double dbl); - перегрузка оператора + для Fraction + double
    - Fraction operator + (int i); - перегрузка оператора + для Fraction + int
    - Fraction& operator += (Fraction fr); - перегрузка оператора += для Fraction += Fraction
    - Fraction& operator += (double dbl); - перегрузка оператора += для Fraction += double
    - Fraction& operator += (int i); - перегрузка оператора += для Fraction += int
    - Fraction& operator = (double dbl); - перегрузка оператора = для Fraction = double
    - Fraction& operator = (const char \*str); - перегрузка оператора = для Fraction = char \*
  + Дружественные функции:
    - friend Fraction operator + (int i, Fraction fr); - перегрузка оператора + для int + Fraction
    - friend Fraction operator + (double dbl, Fraction fr); - перегрузка оператора + для double + Fraction
    - friend std::istream& operator >> (std::istream &in, Fraction &fr); - перегрузка ввода дроби
    - friend std::ostream& operator << (std::ostream &out, const Fraction &fr); - перегрузка вывода дроби

Функции в main.cpp:

* int demo() - демонстрационный режим
* int interactive() - интерактивный режим

**Текст программы**

Main.cpp

#include "fraction/fraction.h"

int demo() {

Fraction z(1, 3);

std::cout << "z=" << z << std::endl;

Fraction fr1(10, 14), fr2;

std::cout << "fr2=" << fr2 << std::endl;

std::cout << "fr1=" << fr1 << std::endl;

Fraction fr = "-1 4/8";

std::cout << "fr=" << fr << std::endl;

Fraction x(z), y;

std::cout << "x=" << x << std::endl;

double dbl = -1.25;

Fraction f = dbl;

std::cout << "f=" << f << std::endl;

y = x + z;

std::cout << "y=" << y << std::endl;

y += x;

f += dbl / 2;

std::cout << "f=" << f << std::endl;

y = x + dbl;

std::cout << "y=" << y << std::endl;

y = dbl + y;

std::cout << "y=" << y << std::endl;

y += dbl;

std::cout << "y=" << y << std::endl;

int i = 5;

y += i;

std::cout << "y=" << y << std::endl;

y = i + x;

std::cout << "y=" << y << std::endl;

y = x + i;

std::cout << "y=" << y << std::endl;

y += dbl + i + x;

std::cout << "y=" << y << std::endl;

return 0;

}

int interactive() {

std::cout << "Введите дробь: \n";

Fraction z;

std::cin >> z;

std::cout << "z=" << z << std::endl;

Fraction fr1(10, 14), fr2;

std::cout << "fr2=" << fr2 << std::endl;

std::cout << "fr1=" << fr1 << std::endl;

Fraction fr = "-1 4/8";

std::cout << "fr=" << fr << std::endl;

Fraction x(z), y;

std::cout << "x=" << x << std::endl;

double dbl = -1.25;

Fraction f = dbl;

std::cout << "f=" << f << std::endl;

y = x + z;

std::cout << "y=" << y << std::endl;

y += x;

f += dbl / 2;

std::cout << "f=" << f << std::endl;

y = x + dbl;

std::cout << "y=" << y << std::endl;

y = dbl + y;

std::cout << "y=" << y << std::endl;

y += dbl;

std::cout << "y=" << y << std::endl;

int i = 5;

y += i;

std::cout << "y=" << y << std::endl;

y = i + x;

std::cout << "y=" << y << std::endl;

y = x + i;

std::cout << "y=" << y << std::endl;

y += dbl + i + x;

std::cout << "y=" << y << std::endl;

return 0;

}

int main(int argc, char \*\*argv) {

if (argc != 2) return 0;

if (!strcmp(argv[1], "demo")) {

demo();

}

else if (!strcmp(argv[1], "int")) {

interactive();

}

return 0;

}

CMakeLists.txt проекта

cmake\_minimum\_required(VERSION 3.23)

set(project "lab03")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(${project}\_SOURCES

main.cpp)

add\_subdirectory(fraction)

set(${project}\_SOURCE\_LIST

${${project}\_SOURCES})

add\_executable(${project}

${${project}\_SOURCE\_LIST})

target\_link\_libraries(${project} fraction)

fraction.h

#pragma once

#include <iostream>

int gcd(int a, int b);

class Fraction {

int numerator;

int denumerator;

public:

void simplify();

Fraction();

Fraction(int \_numerator, int \_denumerator);

Fraction(const char \*str);

Fraction(const Fraction &fr);

Fraction(double dbl);

int getNumerator();

void setNumerator(int \_numerator);

int getDenumerator();

void setDenumerator(int \_denumerator);

Fraction operator + (const Fraction &fr);

Fraction operator + (double dbl);

Fraction operator + (int i);

Fraction& operator += (const Fraction &fr);

Fraction& operator += (double dbl);

Fraction& operator += (int i);

Fraction& operator = (double dbl);

Fraction& operator = (const char \*str);

friend Fraction operator + (int i, const Fraction &fr);

friend Fraction operator + (double dbl, const Fraction &fr);

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

friend std::ostream& operator << (std::ostream &out, const Fraction &fr);

};

fracion.cpp

#include "fraction.h"

int gcd(int a, int b) {

return b ? gcd(b, a % b) : a;

}

void Fraction::simplify() {

int \_gcd = gcd(numerator, denumerator);

numerator /= \_gcd;

denumerator /= \_gcd;

if (denumerator < 0) {

numerator \*= -1;

denumerator \*= -1;

}

}

Fraction::Fraction() {

numerator = 0;

denumerator = 1;

}

Fraction::Fraction(int \_numerator, int \_denumerator) {

if (\_denumerator == 0) {

std::cout << "Dividing by zero\n";

exit(0);

}

numerator = \_numerator;

denumerator = \_denumerator;

simplify();

}

Fraction::Fraction(const char \*str) {

int int\_part = 0, num = 0, denum = 1, len = strlen(str);

int space\_ind = -1, slash\_ind = -1;

for (int i = 0; i < len - 1; i++) {

if (str[i] == ' ' && str[i + 1] != ' ') {

space\_ind = i;

char int\_part\_a[i + 1];

for (int j = 0; j < i; j++) {

int\_part\_a[j] = str[j];

}

int\_part\_a[i] = '\0';

int\_part = atoi(int\_part\_a);

break;

}

}

for (int i = space\_ind + 1; i < len; i++) {

if (str[i] == '/') {

slash\_ind = i;

char num\_a[len - space\_ind + 1], denum\_a[len - space\_ind + 1];

for (int j = space\_ind + 1; j < i; j++) {

num\_a[j - space\_ind - 1] = str[j];

}

num\_a[i - space\_ind - 1] = '\0';

for (int j = i + 1; j < len; j++) {

denum\_a[j - i - 1] = str[j];

}

denum\_a[len - i - 1] = '\0';

num = atoi(num\_a);

denum = atoi(denum\_a);

break;

}

}

if (slash\_ind == -1) {

num = atoi(str);

}

if (int\_part < 0) num = int\_part \* denum - num;

else num = int\_part \* denum + num;

if (denum == 0) {

std::cout << "Dividing by zero\n";

exit(0);

}

numerator = num;

denumerator = denum;

simplify();

}

Fraction::Fraction(const Fraction &fr) {

numerator = fr.numerator;

denumerator = fr.denumerator;

}

Fraction::Fraction(double dbl) {

double i = std::floor(dbl);

double frac = dbl - i;

int acc = 1000000000;

denumerator = acc;

numerator = round(frac \* acc);

simplify();

numerator += i \* denumerator;

}

int Fraction::getNumerator() {

return numerator;

}

void Fraction::setNumerator(int \_numerator) {

numerator = \_numerator;

simplify();

}

int Fraction::getDenumerator() {

return denumerator;

}

void Fraction::setDenumerator(int \_denumerator) {

denumerator = \_denumerator;

simplify();

}

Fraction Fraction::operator + (const Fraction &fr) {

int new\_denum = denumerator \* fr.denumerator;

int new\_num = numerator \* fr.denumerator + fr.numerator \* denumerator;

return Fraction(new\_num, new\_denum);

}

Fraction Fraction::operator + (double dbl) {

Fraction fr = dbl;

return \*this + fr;

}

Fraction Fraction::operator + (int i) {

return Fraction(numerator + i \* denumerator, denumerator);

}

Fraction& Fraction::operator += (const Fraction &fr) {

\*this = \*this + fr;

return \*this;

}

Fraction& Fraction::operator += (double dbl) {

\*this = \*this + dbl;

return \*this;

}

Fraction& Fraction::operator += (int i) {

\*this = \*this + i;

return \*this;

}

Fraction& Fraction::operator = (double dbl) {

\*this = Fraction(dbl);

return \*this;

}

Fraction& Fraction::operator = (const char \*str) {

\*this = Fraction(str);

return \*this;

}

Fraction operator + (int i, const Fraction &fr) {

return Fraction(i \* fr.denumerator + fr.numerator, fr.denumerator);

}

Fraction operator + (double dbl, const Fraction &fr) {

Fraction fr\_left = dbl;

return fr\_left + fr;

}

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

char str[50];

in.getline(str, 50);

fr = str;

return in;

}

std::ostream& operator << (std::ostream &out, const Fraction &fr) {

int int\_part = fr.numerator / fr.denumerator;

int new\_num = fr.numerator - int\_part \* fr.denumerator;

if (int\_part != 0) out << int\_part << ' ';

if (new\_num != 0) out << new\_num \* (int\_part >= 0 ? 1 : -1) << '/' << fr.denumerator;

if (int\_part == 0 && new\_num == 0) out << 0;

return out;

}

CMakeLists.txt библиотеки fraction

cmake\_minimum\_required(VERSION 3.23)

set(project "fraction")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(${project}\_SOURCES

fraction.cpp)

set(${project}\_HEADERS

fraction.h)

set(${project}\_SOURCE\_LIST

${${project}\_SOURCES}

${${project}\_HEADERS})

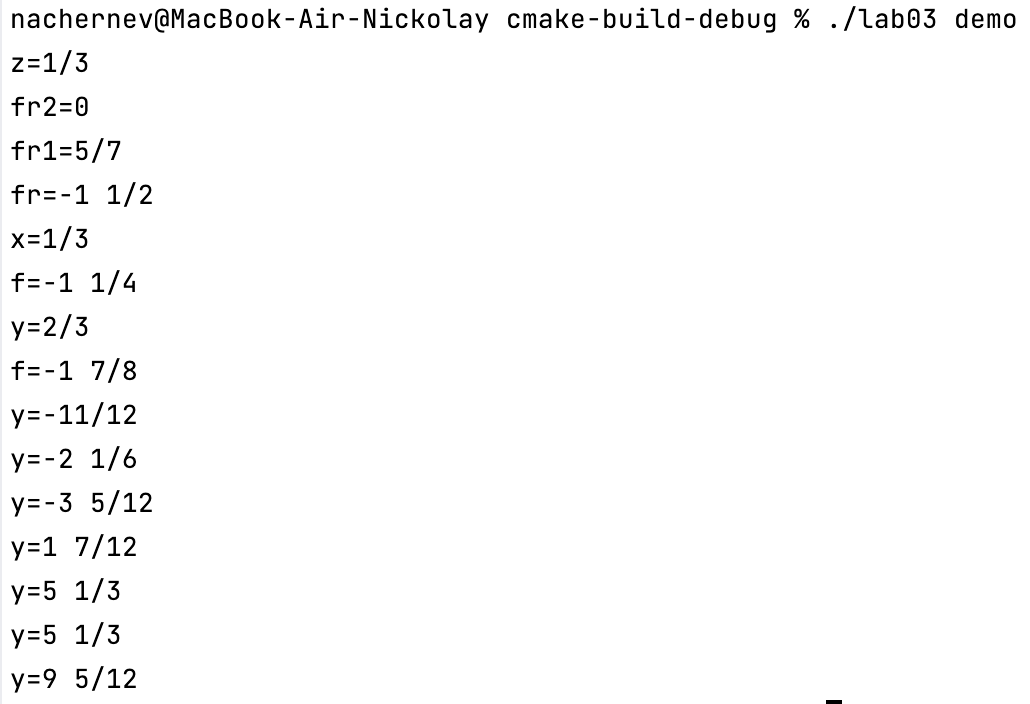
add\_library(${project}

STATIC

${${project}\_SOURCE\_LIST})

**Анализ результатов**

Демонстрационный режим:



Интерактивный режим:

