**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 ИУ5-22Б |  | Преподаватель каф. ИУ5 |
| Поляков Л. С. |  | Бурмистрова М.В. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

# Постановка задачи

**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/)).

**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.

**Требования:**

* Класс "Дроби" необходимо упаковать в самостоятельную статическую библиотеку.
* В работе запрещено использование типа string для обработки ввода с клавиатуры. Необходимо использовать char\*.

# Разработка алгоритма

## Структура проекта

.

├── CMakeLists.txt

├── Fraction – статическая библиотека с классом Fraction

│   ├── CMakeLists.txt

│   ├── Fraction.cpp

│   └── Fraction.hpp

└── main.cpp

## Текст программы

**Fraction.hpp**

#ifndef LAB\_3\_FRACTION\_HPP  
#define LAB\_3\_FRACTION\_HPP  
  
#include <iostream>  
#include <iomanip>  
#include <locale>  
#include <cmath>  
#include <cstring>  
  
  
static int N\_DEC = 4;  
  
class Fraction {  
private:  
 int numerator;  
 int denominator;  
public:  
 Fraction();  
 Fraction(int, int);  
 Fraction(double);  
 Fraction(char\*);  
  
 friend std::ostream& operator << (std::ostream&, const Fraction&);  
 friend std::istream& operator >> (std::istream&, Fraction&);  
 friend Fraction operator + (Fraction&, Fraction&);  
// Fraction operator + (Fraction&);  
 friend Fraction operator + (Fraction&, int);  
 friend Fraction operator + (int, Fraction&);  
 friend Fraction operator += (Fraction&, Fraction&);  
 friend Fraction operator += (Fraction&, int);  
 friend Fraction operator + (Fraction&, double);  
 friend Fraction operator + (double, Fraction&);  
 friend Fraction operator += (Fraction&, double);  
 void toIrreducible();  
 int getDenominator();  
};  
  
int countDigits(double);  
int NOD(int, int);  
int NOK(int, int);  
  
#endif //LAB\_3\_FRACTION\_HPP

**Fraction.cpp**

#include "Fraction.hpp"  
  
Fraction::Fraction() {  
 numerator = 0;  
 denominator = 1;  
}  
  
Fraction::Fraction(int \_numerator, int \_denominator) : numerator(\_numerator), denominator(\_denominator) {  
 toIrreducible();  
}  
  
Fraction::Fraction(double num) {  
 denominator = 1;  
 int temp = countDigits(num);  
 numerator = num \* pow(10, temp);  
 denominator \*= pow(10, temp);  
 toIrreducible();  
}  
  
Fraction::Fraction(char\* num) {  
 int fullPart;  
 char\* pEnd, \*pch, \*temp;  
 pch = std::strchr(num, '/');  
 temp = std::strchr(num, ' ');  
 // with whole part  
 if (temp != nullptr) {  
 fullPart = strtol(num, &pEnd, 10);  
 numerator = strtol(pEnd, &pch, 10);  
 denominator = strtol(pch + 1, nullptr, 10);  
 if ((fullPart < 0) || (num[0] == '-'))  
 numerator \*= -1;  
 numerator += fullPart \* denominator;  
 }  
 // without fractional part  
 else if (pch == nullptr) {  
 numerator = strtol(num, nullptr, 10);  
 denominator = 1;  
 }  
 // only fraction  
 else {  
 numerator = strtol(num, &pch, 10);  
 denominator = strtol(pch + 1, nullptr, 10);  
 }  
 toIrreducible();  
}  
  
void Fraction::toIrreducible() {  
 if (this->denominator != 0 and this->numerator != 0) {  
 int nod = NOD(abs(this->numerator), abs(this->denominator));  
 this->numerator /= nod;  
 this->denominator /= nod;  
 }  
  
}  
  
int Fraction::getDenominator()  
{  
 return this->denominator;  
}  
  
std::ostream& operator<<(std::ostream& out, const Fraction& fraction) {  
 if (fraction.denominator == 1 && fraction.numerator != 0)  
 out << fraction.numerator;  
 else if (fraction.numerator == 0)  
 out << 0;  
 else if (fraction.denominator == 0)  
 std::cout << "Error: zero in denominator" << std::endl;  
 else {  
 if (abs(fraction.numerator) > fraction.denominator)  
 std::cout << fraction.numerator / fraction.denominator << ' ' << abs(fraction.numerator) % fraction.denominator << '/' << fraction.denominator;  
 else  
 out << fraction.numerator << '/' << fraction.denominator;  
 }  
 return out;  
}  
  
std::istream& operator>>(std::istream& in, Fraction& fraction) {  
 char\* line = new char[60];  
 std::cin.getline(line, 60, '\n');  
 fraction = Fraction(line);  
 return in;  
}  
  
Fraction operator+(Fraction& lhs, Fraction& rhs) {  
 rhs.toIrreducible();  
 lhs.toIrreducible();  
 int nok = NOK(lhs.denominator, rhs.denominator);  
 Fraction result((nok / lhs.denominator) \* lhs.numerator + (nok / rhs.denominator) \* rhs.numerator, nok);  
 result.toIrreducible();  
 return result;  
}  
  
//Fraction Fraction::operator+(Fraction &rhs) {  
// int nok = NOK(denominator, rhs.denominator);  
// Fraction result((nok / denominator) \* numerator + (nok / rhs.denominator) \* rhs.numerator, nok);  
// result.toIrreducible();  
// return result;  
//}  
  
Fraction operator+(Fraction& lhs, const int rhs) {  
 Fraction result(lhs.numerator + rhs \* lhs.denominator, lhs.denominator);  
 result.toIrreducible();  
 return result;  
}  
  
Fraction operator+(const int lhs, Fraction& rhs) {  
 Fraction result(rhs.numerator + lhs \* rhs.denominator, rhs.denominator);  
 result.toIrreducible();  
 return result;  
}  
  
Fraction operator+=(Fraction& lhs, Fraction& rhs) {  
 lhs = rhs + lhs;  
 lhs.toIrreducible();  
 return lhs;  
}  
  
Fraction operator+=(Fraction& lhs, const int rhs) {  
 lhs.numerator += rhs \* lhs.denominator;  
 lhs.toIrreducible();  
 return lhs;  
}  
  
Fraction operator+(Fraction& lhs, const double rhs) {  
 Fraction result(rhs);  
 result += lhs;  
 result.toIrreducible();  
 return result;  
}  
  
Fraction operator+(const double lhs, Fraction& rhs) {  
 Fraction result(lhs);  
 result += rhs;  
 result.toIrreducible();  
 return result;  
}  
  
Fraction operator+=(Fraction& lhs, const double rhs) {  
 lhs = lhs + rhs;  
 lhs.toIrreducible();  
 return lhs;  
}  
  
int countDigits(double num) {  
 int count = 0;  
 while (num != int(num)) {  
 num \*= 10;  
 count++;  
 }  
 return count;  
}  
  
int NOD(int a, int b) {  
 if (a < b) {  
 std::swap(a, b);  
 }  
 while (a % b != 0) {  
 a = a % b;  
 std::swap(a, b);  
 }  
 return b;  
}  
  
int NOK(int a, int b) {  
 return a \* b / NOD(a, b);  
}

**Fraction/** **CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.26)  
  
set(PROJECT\_NAME Fraction)  
project(${PROJECT\_NAME})  
  
set(CMAKE\_CXX\_STANDARD 20)  
  
set(${PROJECT\_NAME}\_HEADERS  
 Fraction.hpp  
)  
  
set(${PROJECT\_NAME}\_SOURCES  
 Fraction.cpp  
)  
  
set(${PROJECT\_NAME}\_SOURCE\_LIST  
 ${${PROJECT\_NAME}\_SOURCES}  
 ${${PROJECT\_NAME}\_HEADERS}  
)  
  
add\_library(${PROJECT\_NAME} STATIC ${${PROJECT\_NAME}\_SOURCE\_LIST})

**main.cpp**

#include <iostream>  
#include "Fraction/Fraction.hpp"  
  
int main() {  
 std::cout << "Enter the fraction: \n";  
 Fraction z;  
 std::cin >> z;  
 if (z.getDenominator() == 0){  
 std::cout << "Error: zero in denominator" << std::endl;  
 return 0;  
 }  
 std::cout << "z=" << z << std::endl;  
 // Constructors check  
 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;  
 // Check of the overloaded operation "+"  
 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;  
 double a = dbl + i;  
 Fraction b = a + x;  
 y += b;  
 std::cout << "y=" << y << std::endl;  
 return 0;  
}

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.26)  
  
set(PROJECT\_NAME Lab\_3)  
project(${PROJECT\_NAME})  
  
set(CMAKE\_CXX\_STANDARD 20)  
  
set(${PROJECT\_NAME}\_HEADERS  
  
)  
  
set(${PROJECT\_NAME}\_SOURCES  
 main.cpp  
)  
  
add\_subdirectory(Fraction)  
  
set(${PROJECT\_NAME}\_SOURCE\_LIST  
 ${${PROJECT\_NAME}\_SOURCES}  
 ${${PROJECT\_NAME}\_HEADERS}  
)  
  
add\_executable(${PROJECT\_NAME} ${${PROJECT\_NAME}\_SOURCE\_LIST})  
  
target\_link\_libraries(${PROJECT\_NAME} Fraction)

# Анализ Результатов

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

# Вывод

Я научился

* Перегружать операторы через методы класса
* Перегружать операторы через дружественные функции класса
* Работать со статическими библиотеками с использованием Cmake