**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 |
| Байдаков В.М. |  |  |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

**Постановка задачи**

Лабораторная работа состоит из двух частей:

Часть 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.Разработать конструктор по умолчанию и конструктор, который преобразует строку, содержащую внешнее представление дроби, в объект класса "Дроби".Перегрузить операции потокового ввода дроби с клавиатуры и вывода ее на экран монитора. При вводе выполнять сокращение дроби.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).

Часть 2:

Дополнить класс функциями-членами класса и функциями-друзьями класса, которые необходимы для выполнения программы из Приложения 1.Выполнить программу из Приложения 1 и сравнить результаты с тестовым примером.

**Разработка алгоритма**

Члены класса:

int numerator; - Числитель дроби.

int denominator; - Знаменатель дроби.

Методы класса:

int nod(int a, int b); - Возвращает наибольший общий делитель (НОД) двух целых чисел a и b, используется для сокращения дроби.

void reduceFraction(); - Сокращает дробь, делив числитель и знаменатель на их НОД и корректирует знаки, чтобы знаменатель был положительным.

Fraction(const int num = 0, const int denom = 1); - Инициализирует дробь заданными значениями числителя и знаменателя, по умолчанию создает дробь 0/1.

Fraction(const double value); - Инициализирует дробь вещественным числом, конвертируя его в дробное представление.

Fraction(const char \*input); - Инициализирует дробь из строкового представления, которое может включать целую часть, числитель и знаменатель.

Fraction operator+(const int &num); - Реализует операцию сложения дроби и целого числа, возвращая новую дробь.

Fraction operator+(const Fraction &value); - Реализует операцию сложения двух дробей, возвращая новую дробь.

Fraction operator+(const double value); - Реализует операцию сложения дроби и вещественного числа, возвращая новую дробь.

Fraction &operator+=(const Fraction &value); - Реализует операцию сложения и присваивания для двух дробей, изменяя текущую дробь.

Fraction &operator+=(const int value); - Реализует операцию сложения и присваивания дроби и целого числа, изменяя текущую дробь.

Fraction &operator+=(const double value); - Реализует операцию сложения и присваивания дроби и вещественного числа, изменяя текущую дробь.

**Прочее:**

std::ostream &operator<<(std::ostream &out, Fraction &f); - Перегружает оператор вывода, позволяя выводить объекты Fraction в стандартные потоки вывода.

std::istream &operator>>(std::istream &in, Fraction &f); - Перегружает оператор ввода для чтения объектов Fraction из стандартных потоков ввода.

Fraction operator+(const Fraction &f, const int value); - Реализует операцию сложения между объектом Fraction и целым числом, возвращая новый объект Fraction, представляющий сумму.

Fraction operator+(const double value, const Fraction &f); - Реализует операцию сложения между вещественным числом и объектом Fraction, возвращая новый объект Fraction, представляющий сумму.

Функции в main.cpp:

Int demo() – демонстрационный режим, где значение дроби z уже введено

Int interactive() - интерактивный режим, где пользователь сам вводит значение дроби z

**Текст программы**

CMakeLists.txt

cmake\_minimum\_required(VERSION 3.23)

set(project "lab3")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(SOURCES

    main.cpp

)

add\_subdirectory(fraction)

add\_executable(${project} ${SOURCES})

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

target\_include\_directories(

    ${PROJECT\_NAME} PUBLIC

    ${CMAKE\_SOURCE\_DIR}/fraction

)

Fraction.h

#pragma once

#include <iostream>

#include <cstring>

#include <cstdlib>

#include <cmath>

class Fraction

{

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

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

    int numerator;

    int denominator;

    int nod(int a, int b)

    {

        return b == 0 ? a : nod(b, a % b);

    }

    void reduceFraction()

    {

        int n = nod(numerator, denominator);

        numerator /= n;

        denominator /= n;

        if (denominator < 0)

        {

            numerator \*= -1;

            denominator \*= -1;

        }

    }

public:

    Fraction(const int num = 0, const int denom = 1)

    {

        numerator = num;

        denominator = denom;

        reduceFraction();

    }

    Fraction(const double value)

    {

        double exponenta = std::floor(value);

        double mantissa = value - exponenta;

        int accuracy = 10e6;

        denominator = accuracy;

        numerator = exponenta \* denominator + round(mantissa \* accuracy);

        reduceFraction();

    }

    Fraction(const char \*input)

    {

        int spaceIndex = -1, slashIndex = -1, size = strlen(input);

        int wholeNum = 0;

        int newNum = 0, newDenom = 1;

        for (size\_t i = 0; i < size; ++i)

        {

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

            {

                spaceIndex = i;

                break;

            }

        }

        if (spaceIndex != -1)

        {

            char wholeNumChar[spaceIndex + 2];

            strncpy(wholeNumChar, input, spaceIndex);

            wholeNumChar[spaceIndex] = '\0';

            wholeNum = atoi(wholeNumChar);

        }

        for (size\_t i = spaceIndex + 1; i < size; ++i)

        {

            if (input[i] == '/')

            {

                slashIndex = i;

                break;

            }

        }

        if (slashIndex != -1)

        {

            char numChar[slashIndex - spaceIndex];

            strncpy(numChar, input + spaceIndex + 1, slashIndex - spaceIndex - 1);

            numChar[slashIndex - spaceIndex - 1] = '\0';

            char denomChar[size - slashIndex];

            strncpy(denomChar, input + slashIndex + 1, size - slashIndex - 1);

            denomChar[size - slashIndex - 1] = '\0';

            newNum = atoi(numChar);

            newDenom = atoi(denomChar);

        }

        else

        {

            newNum = atoi(input);

            denominator = 1;

        }

        newNum = wholeNum \* newDenom + (wholeNum < 0 ? -newNum : newNum);

        numerator = newNum;

        denominator = newDenom;

        reduceFraction();

    }

    Fraction operator+(const int &num)

    {

        return Fraction{numerator + num \* denominator, denominator};

    }

    Fraction operator+(const Fraction &value)

    {

        int newDenom = denominator \* value.denominator;

        int newNum = numerator \* value.denominator + value.numerator \* denominator;

        return Fraction(newNum, newDenom);

    }

    Fraction operator+(const double value)

    {

        Fraction a = value;

        return a + \*this;

    }

    Fraction &operator+=(const Fraction &value)

    {

        Fraction a = \*this + value;

        return \*this = a;

    }

    Fraction &operator+=(const int value)

    {

        Fraction a = \*this + value;

        return \*this = a;

    }

    Fraction &operator+=(const double value)

    {

        Fraction a = \*this + value;

        return \*this = a;

    }

    friend Fraction operator+(const Fraction &f, const int value);

    friend Fraction operator+(const double value, const Fraction &f);

};

std::ostream &operator<<(std::ostream &out, Fraction &f);

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

fraction.cpp

#include "fraction.h"

std::ostream &operator<<(std::ostream &out, Fraction &f)

{

    if (f.numerator == 0)

    {

        out << 0;

        return out;

    }

    f.reduceFraction();

    if (f.denominator == 1)

    {

        out << f.numerator;

        return out;

    }

    if (abs(f.numerator) > f.denominator)

    {

        int num = f.numerator / f.denominator;

        if (f.numerator - num \* f.denominator == 0)

        {

            out << num;

            return out;

        }

        out << num << ' ' << abs(f.numerator - num \* f.denominator) << '/' << f.denominator;

    }

    else

    {

        out << f.numerator << '/' << f.denominator;

    }

    return out;

}

std::istream &operator>>(std::istream &in, Fraction &f)

{

    char str[50];

    in.getline(str, 50);

    f = str;

    if (f.denominator == 0)

    {

        std::cerr<<"CAN'T DIVIDE BY ZERO";

    }

    return in;

}

Fraction operator+(const Fraction& f, const int value){

    Fraction a(f.denominator\*value+f.numerator,f.denominator);

    return a;

}

Fraction operator+(const double value, const Fraction& f){

    Fraction a = value;

    return a + f;

}

Main.cpp

#include "Fraction.h"

int interactive(){

*// ввод дроби с клавиатуры*

    std::cout << "Enter Fraction values: \n";

    Fraction z;

    std::cin >> z;

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

*// проверка конструкторов*

    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;

*// проверка перегруженной операции "+"*

*//std::cout<<"heue\n";*

    y = x + z;

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

    y += x;*//vsrreg*

    f += (dbl / 2);

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

*//std::cout<<x<<"\n";*

    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 << "x= " << x << std::endl;*

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

    y = x + i;

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

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

*//std::cout<<y<<" "<<dbl<<" "<<" "<<x<<" "<<i<<"\n";*

    y += dbl  + x + i;

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

    return 0;

}

int demo(){

*// ввод дроби с клавиатуры*

    std::cout << "Fraction z equals -1 23/467 \n";

    Fraction z{"-1 23/467"};

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

*// проверка конструкторов*

    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;

*// проверка перегруженной операции "+"*

    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[]) {

    system("chcp 1251");

    bool isInteractive = false; *// по умолчанию демо-режим.*

*// Если i передается в качестве аргумента, то программу*

*// необходимо запустить в интерактивном режиме*

    if ((argc == 2) && strcmp(argv[1], "i") == 0) {

        isInteractive = true;

    }

    if (isInteractive) {

        return interactive();

    } else {

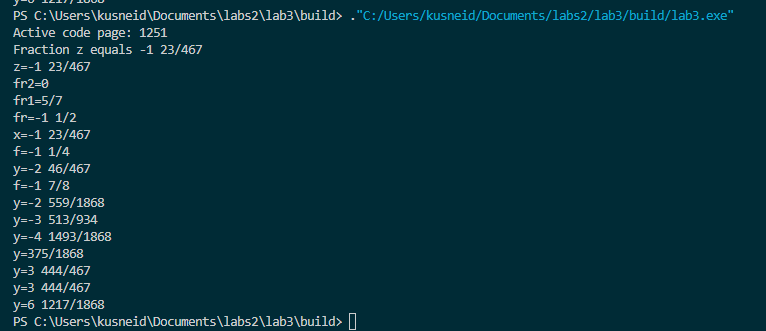
        return demo();

    }

}

Анализ результатов

Демо-режим:



Интерактивный режим

