**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №3

«Классы. Перегрузка конструкторов и операций»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-21 |  | преподаватель каф. ИУ5 |
| Агличеев Михаил |  | Селиверстова А.В. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

1. Постановка задачи.

Создать класс для работы с обыкновенными дробями. Все операции, которые должны

выполняться с дробями, включены в программу в приложении 1.

Числитель и знаменатель дроби имеют тип int.

Дроби вводятся как строка, имеющая вид:

- для дробей с целой частью: знак, целая часть, пробел, числитель, слэш (‘/’), знаменатель,. например: -2

6/18, 5 9/3, 2 4/1.

- для дробей без целой части: знак, числитель, слэш (‘/’), знаменатель,

например: 3/4, -9/3, -8/6 (знаменатель всегда положительный).

Значения представленных выше дробей на экране при выводе должны иметь вид:

-2 1/3, 8, 6.

3/4, -3, -1 1/3.

При выводе и после выполнения арифметических операций дроби сокращаются, то есть числитель и

знаменатель не должны иметь общих множителей.

Перегрузить операции &#39;+&#39;, &#39;+=&#39; для сложения дробей и дроби и целого в любых сочетаниях (дробь+целое,

целое+дробь, дробь+дробь).

Перегрузить операции &#39;+&#39;, &#39;+=&#39; для сложения дроби и double в любых сочетаниях (дробь+double,

double+дробь). Преобразование double-дробь должно выполняться с точностью до N\_DEC десятичных

знаков после запятой, где N\_DEC – целочисленная константа, задаваемая пользователем. Задайте значение

по умолчанию N\_DEC=4.

Для инициализации объектов разрабатываемого класса обыкновенных дробей предусмотреть

соответствующие конструкторы (с одним аргументом типа char\*, с одним аргументом типа double и с

двумя аргументами типа int, которые имеют значения по умолчанию).

При перегрузке операций использовать функции - члены класса, а где это невозможно, то функции -

друзья класса.

Для обеспечения более удобного контроля результатов выполнения программы вставьте в конструкторы и

перегруженные операции операторы вывода, идентифицирующие выполняемую функцию.

Выполните следующий эксперимент: закомментируйте операции дроби с int и повторно выполните

программу. Объясните результаты сложения дробей с целыми числами.

1. Разработка Интерфейса класса

Поля класса Fractions:

Private: char\* representation – Представление дроби в виде строки. (Например, “3/5”)

int numerator – Числитель дроби

int denominator – Знаменатель дроби (может быть лишь положительным)

Методы Класса Fractions:

Private: void normalize() – нормализует дробь (делит числитель и знаменатель на их НОД)

void update\_representation – приводит текстовое представление дроби в соответствие с её значением

Public: Fractions(int new\_numerator = 0, int new\_denominator = 1) – конструктор дроби из числителя и знаменателя, одного целого числа и конструктор по умолчанию

Fractions(char\*) - конструктор дроби из её строкового представления

Fractions(double, int N\_DEC = 4);

~Fractions();

void change\_numden(int, int) – Изменение числителя и знаменателя на передаваемые в качестве аргументов значения

void change\_representation(char\*) – Изменение строкового представления на передаваемое в качестве аргумента

friend ostream& operator<< (ostream&, const Fractions&) – вывод дроби в поток

friend istream& operator>> (istream&, Fractions&) – ввод дроби из потока

void operator= (const Fractions&) – копирование дроби

Fractions operator+(Fractions&) – Здесь и далее – математические операции для сложения дроби с дробью, числом с плавающей точкой и целым, в обоих возможных порядках

Fractions operator+(double);

Fractions operator+(int);

friend Fractions operator+(double, Fractions&);

friend Fractions operator+(int, Fractions&);

void operator+=(Fractions&);

void operator+=(double);

void operator+=(int);

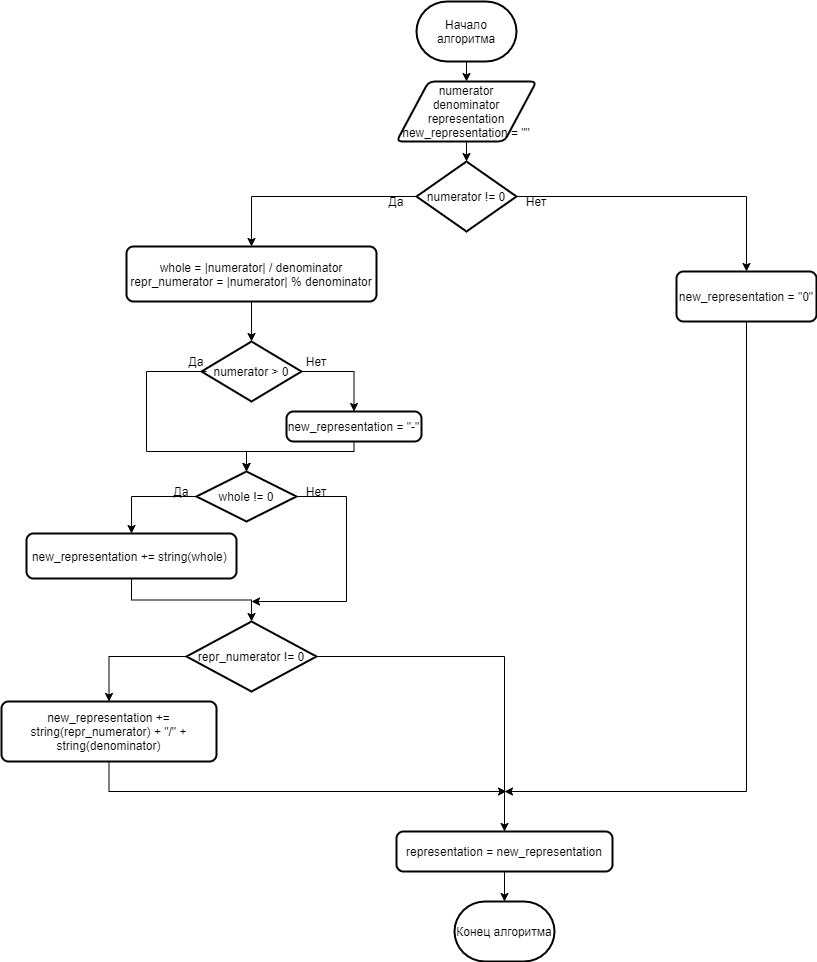
friend void operator+=(double&, Fractions&);

friend void operator+=(int&, Fractions&);

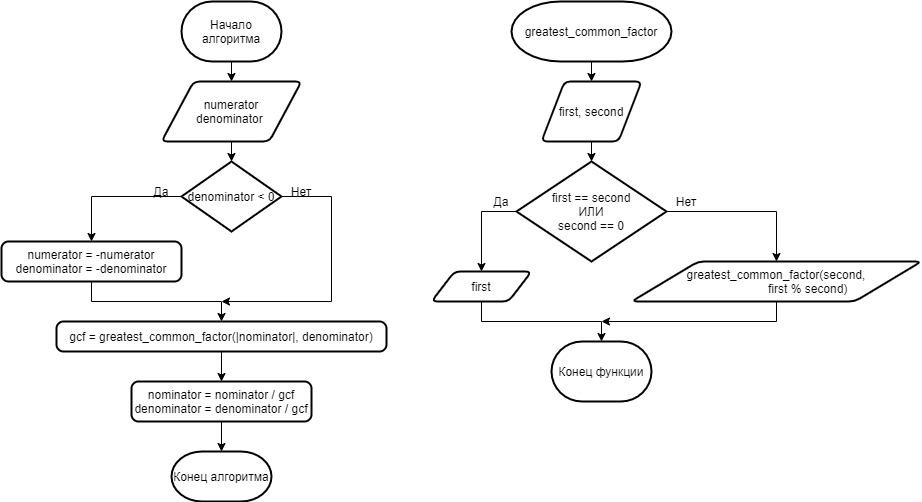
|  |  |
| --- | --- |
|  |  |

1. Разработка алгоритма

Блок-схема алгоритма вывода дробей (но, т.к. их вывод представляет собой просто вывод на экран текстового представления, здесь я покажу, как из пары числитель-знаменатель получается текстовое представление дроби)



Блок-схема алгоритма сокращения дробей:



1. Текст программы

Файл fraction.h:

#ifndef \_\_FRACTION\_H\_\_

#define \_\_FRACTION\_H\_\_

#include <iostream>

using namespace std;

class Fractions {

private:

char\* representation;

int numerator;

int denominator;

void normalise();

void update\_representation();

public:

Fractions(int new\_numerator = 0, int new\_denominator = 1);

Fractions(char\*);

Fractions(double, int N\_DEC = 4);

~Fractions();

void change\_numden(int, int);

void change\_representation(char\*);

friend ostream& operator<< (ostream&, const Fractions&);

friend istream& operator>> (istream&, Fractions&);

void operator= (const Fractions&);

Fractions operator+(Fractions&);

Fractions operator+(double);

Fractions operator+(int);

friend Fractions operator+(double, Fractions&);

friend Fractions operator+(int, Fractions&);

void operator+=(Fractions&);

void operator+=(double);

void operator+=(int);

friend void operator+=(double&, Fractions&);

friend void operator+=(int&, Fractions&);

};

int chars\_to\_int(char\*, int, int);

char\* ints\_to\_char(char\*, int);

char\* cut(char\*);

int greatest\_common\_factor(int, int);

double round(double);

#endif //\_\_FRACTION\_H\_\_

Файл fraction.cpp:

#include <cmath>

#include "fraction.h"

Fractions::Fractions(int new\_numerator, int new\_denominator) {

representation = cut("1");

denominator = 1;

change\_numden(new\_numerator, new\_denominator);

}

Fractions::Fractions(char\* new\_representation) {

representation = cut("1");

numerator = 1; denominator = 1;

change\_representation(new\_representation);

}

Fractions::Fractions(double new\_double, int N\_DEC) {

representation = cut("1");

numerator = round(new\_double \* pow(10, N\_DEC));

denominator = pow(10, N\_DEC);

normalise();

update\_representation();

}

Fractions::~Fractions() {

if (representation == nullptr) {

delete[] representation;

}

}

void Fractions::change\_numden(int new\_numerator, int new\_denominator) {

numerator = new\_numerator;

if (denominator) {

denominator = new\_denominator;

}

else {

cout << "Denominator can't be 0! Leaving denominator as it is (1 by default)." << endl;

}

normalise();

update\_representation();

}

void Fractions::change\_representation(char\* new\_representation) {

int space\_pos = -1;

int slash\_pos = -1;

int i, whole = 0, new\_numerator = 0, new\_denominator = 1;

char character = ' ';

bool negative = false;

if (new\_representation[0] == '-') {

negative = true;

new\_representation++;

}

for (i = 0; new\_representation[i]; i++) { // Checking for disallowed symbols.

character = new\_representation[i];

if ((character != ' ') && (character < '/' || character > '9')) {

cout << "Disallowed symbols! Representation remains unchanged ('1' by default)." << endl;

return;

}

}

if (new\_representation[0] == ' ' || character == ' ' ||

new\_representation[0] == '/' || character == '/') {

cout << "Error when parsing end of representation! Representation remains unchanged ('1' by default)." << endl;

return;

}

for (i = 1; i < strlen(new\_representation); i++) { // Finding where space(s) and slash(es) are.

if (new\_representation[i] == ' ') {

if (space\_pos != -1) {

cout << "There can't be 2 spaces! Representation remains unchanged ('1' by default)." << endl;

return;

}

space\_pos = i;

}

else if (new\_representation[i] == '/') {

if (slash\_pos != -1) {

cout << "There can't be 2 slashes! Representation remains unchanged ('1' by default)." << endl;

return;

}

slash\_pos = i;

}

}

if (space\_pos != -1) { // In this case we have both whole part and a fraction.

if (slash\_pos < space\_pos) {

cout << "Something is wrong with slash! Representation remains unchanged ('1' by default)." << endl;

return;

}

whole = chars\_to\_int(new\_representation, 0, space\_pos - 1);

new\_numerator = chars\_to\_int(new\_representation, space\_pos + 1, slash\_pos - 1);

new\_denominator = chars\_to\_int(new\_representation, slash\_pos + 1, strlen(new\_representation) - 1);

}

else {

if (slash\_pos != -1) { // Only fraction.

new\_numerator = chars\_to\_int(new\_representation, 0, slash\_pos - 1);

new\_denominator = chars\_to\_int(new\_representation, slash\_pos + 1, strlen(new\_representation) - 1);

}

else { // Only whole part.

whole = chars\_to\_int(new\_representation, 0, strlen(new\_representation) - 1);

}

}

if (!new\_denominator) {

cout << "Denominator can't be 0! Representation remains unchanged ('1' by default)." << endl;

return;

}

denominator = new\_denominator;

numerator = new\_numerator + whole \* denominator;

if (negative) {

numerator \*= -1;

}

normalise();

update\_representation();

}

void Fractions::update\_representation() {

int whole;

int repr\_numerator;

char new\_repr[100];

char\* p\_new\_repr = new\_repr;

if (numerator) {

whole = abs(numerator) / denominator;

repr\_numerator = abs(numerator) % denominator;

if (numerator < 0) {

new\_repr[0] = '-';

p\_new\_repr++;

}

if (whole) {

p\_new\_repr = ints\_to\_char(p\_new\_repr, whole);

p\_new\_repr[0] = ' ';

p\_new\_repr[1] = '\0';

p\_new\_repr++;

}

if (repr\_numerator) {

p\_new\_repr = ints\_to\_char(p\_new\_repr, repr\_numerator);

p\_new\_repr[0] = '/';

p\_new\_repr++;

ints\_to\_char(p\_new\_repr, denominator);

}

}

else {

new\_repr[0] = '0';

new\_repr[1] = '\0';

}

delete[] representation;

representation = cut(new\_repr);

}

void Fractions::normalise() {

if (denominator < 0) {

numerator \*= -1;

denominator \*= -1;

}

int gcf = greatest\_common\_factor(abs(numerator), denominator); // Denominator is already bigger than 0, numerator is not guaranteed to be.

numerator /= gcf;

denominator /= gcf;

}

int chars\_to\_int(char\* input, int start, int end) { // Turn input from start to end inclusive into integer.

int output;

int i;

for (i = start; i < end && input[i] - '0' == 0; i++); // Get index of first non-0 char or end, if all of the chars before ending one are 0.

output = (input[i] - '0');

i++;

for (i; i <= end; i++) {

output = output \* 10 + (input[i] - '0');

}

return output;

}

char\* ints\_to\_char(char\* output, int input) { // Turn int into char\* and return pointer to '\0' character of that char\*.

int i, shift, as\_numbers[100];

if (input != 0) {

for (i = 0; input; i++) {

as\_numbers[i] = input % 10;

input /= 10;

}

}

else {

i = 1;

as\_numbers[0] = 0;

}

shift = i;

output[i] = '\0';

i--;

for (i; i >= 0; i--) {

output[shift - 1 - i] = as\_numbers[i] + '0';

}

return output + shift;

}

char\* cut(char\* inp) {

char\* result = new char[strlen(inp) + 1];

int i;

for (i = 0; inp[i]; i++) {

result[i] = inp[i];

}

result[i] = inp[i];

return result;

}

int greatest\_common\_factor(int first, int second) { // Euclid's algorythm.

if (first == second || !second) {

return first;

}

return greatest\_common\_factor(second, first % second);

}

double round(double x) {

return floor(x + 0.5);

}

Файл fractionsCheck.h:

#ifndef \_\_FRACTIONSCHECK\_H\_\_

#define \_\_FRACTIONSCHECK\_H\_\_

int main();

#endif //\_\_FRACTIONSCHECK\_H\_\_

Файл fractionsCheck.cpp:

#include <iostream>

#include "fractionsCheck.h"

#include "fraction.h"

#include "windows.h"

using namespace std;

int main() {

setlocale(0, "russian");

//ввод дроби с клавиатуры

cout << "Введите дробь: \n";

Fractions z;

cin >> z;

cout << "z=" << z << endl;

//проверка конструкторов

Fractions fr1(10, 14), fr2;

cout << "fr2=" << fr2 << endl;

cout << "fr1=" << fr1 << endl;

Fractions fr = "-1 4/8";

cout << "fr=" << fr << endl;

Fractions x(z), y;

cout << "x=" << x << endl;

double dbl = -1.25;

Fractions f = dbl;

cout << "f=" << f << endl;

//проверка перегруженной операции "+"

y = x + z;

cout << "y=" << y << endl;

y += x;

f += dbl / 2;

cout << "f=" << f << endl;

y = x + dbl;

cout << "y=" << y << endl;

y = dbl + y;

cout << "y=" << y << endl;

y += dbl;

cout << "y=" << y << endl;

int i = 5;

y += i;

cout << "y=" << y << endl;

y = i + x;

cout << "y=" << y << endl;

y = x + i;

cout << "y=" << y << endl;

y += dbl + i + x;

cout << "y=" << y << endl;

system("pause");

}

5. Анализ результатов

