**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Отчет по лабораторной работе №3

«**Классы. Перегрузка конструкторов и операций.**»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Выполнил:** | |  | **Принял:** | |
| ФИО: | \_Цыпышев Т. А.\_\_\_\_\_ |  | ФИО: | \_Правдина А. Д.\_\_\_\_ |
| Группа: | \_ИУ5-21Б\_\_\_\_\_\_\_\_\_\_ |  | Должность: | \_Преподаватель\_\_\_\_ |
| Дата: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Дата: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва, 2022 г.

**Постановка задачи**

**Цель работы:**

* Создать класс для работы с обыкновенными дробями, который будет иметь функции для выполнения всех операций с дробями, описанных в приложении 1.
* Реализовать возможность ввода дробей в виде строки и вывода дробей в виде строки с учетом требований к формату.
* Реализовать функции-члены класса и функции-друзья класса для перегрузки операций '+', '+=' для сложения дробей и дроби и целого в любых сочетаниях, а также для сложения дроби и double в любых сочетаниях.
* Разработать конструкторы для инициализации объектов разрабатываемого класса обыкновенных дробей.
* Реализовать операторы вывода, идентифицирующие выполняемую функцию, чтобы обеспечить более удобный контроль результатов выполнения программы.

**Разработка алгоритма**

int main()

Функция main() является точкой входа в программу. Она начинает выполнение программы, определяет объекты, вызывает функции и операции, обрабатывает и выводит результаты.

int gcd(int a, int b) {

Функция gcd() реализует алгоритм поиска наибольшего общего делителя (НОД) двух целых чисел "a" и "b".

class fraction

Класс "fraction" представляет дробь и имеет следующие переменные-члены:

* "sign" - логическая переменная, определяющая знак дроби (true, если отрицательная, false, если положительная).
* "integer" - целочисленная переменная, представляющая целую часть дроби.
* "numerator" - целочисленная переменная, представляющая числитель дроби.
* "denominator" - целочисленная переменная, представляющая знаменатель дроби.

friend std::istream &operator>>(std::istream &input, fraction &item)

Этот код определяет перегруженный оператор ">>" в качестве friend-функции для класса "fraction", который позволяет считывать значения переменных-членов объекта класса "fraction" из потока std::istream.

Аргументы функции:

* input - ссылка на объект потока ввода std::istream, из которого происходит считывание значений переменных-членов объекта класса "fraction".
* item - ссылка на объект класса "fraction", для которого происходит считывание значений переменных-членов из потока.

Функция начинает считывание, проверяя знак дроби. Если в потоке есть знак "-", то переменная-член "sign" устанавливается в true (отрицательная дробь), в противном случае - в false (положительная дробь). Далее, функция считывает значение целой части дроби и проверяет, есть ли после нее дробная часть. Если после целой части есть пробел, то функция считывает числитель и знаменатель дроби. Затем функция вычисляет НОД между числителем и знаменателем с помощью функции "gcd" и упрощает дробь. Наконец, функция возвращает ссылку на объект потока ввода std::istream.

friend std::ostream &operator<<(std::ostream &output, const fraction &item)

Этот код определяет перегруженный оператор "<<" в качестве friend-функции для класса "fraction", который позволяет выводить значения переменных-членов объекта класса "fraction" в поток std::ostream.

Аргументы функции:

* output - ссылка на объект потока вывода std::ostream, в который происходит вывод значений переменных-членов объекта класса "fraction".
* item - константная ссылка на объект класса "fraction", значения переменных-членов которого необходимо вывести в поток.

Функция начинает вывод значений, проверяя знак дроби. Если дробь отрицательная (переменная-член "sign" равна true), то в поток выводится знак "-". Далее, функция проверяет, есть ли у дроби целая часть. Если целая часть есть, то она выводится в поток с пробелом после нее. Затем, если числитель дроби не равен нулю, функция выводит его, знак "/" и знаменатель дроби. Наконец, если дробь является нулем (целая часть, числитель и знаменатель равны нулю), то в поток выводится символ "0". Функция возвращает ссылку на объект потока вывода std::ostream.

fraction operator+(const fraction &other)

Эта функция представляет оператор сложения двух дробей (fraction). Она принимает ссылку на другой объект дроби (other) и возвращает новый объект дроби, который является результатом сложения.

Переменные, используемые в функции:

* fraction result: это объект дроби, который будет использоваться для хранения результата сложения.
* int numerator\_: временная переменная, используемая для хранения числителя результатирующей дроби на каждой итерации.
* int denominator\_: временная переменная, используемая для хранения знаменателя результатирующей дроби на каждой итерации.

Функция начинается с проверки специального случая, когда числитель одной или обеих дробей равен нулю. Если обе дроби имеют числитель, то выполняется обычное сложение. На каждой итерации происходит вычисление числителя и знаменателя результирующей дроби в соответствии с правилами сложения дробей. Затем вычисляется целая часть результата и оставшаяся часть дроби, используя арифметику целых чисел. Затем дробь упрощается путем нахождения наибольшего общего делителя числителя и знаменателя, и затем дробь сокращается до несократимой формы. Наконец, результат возвращается в виде нового объекта дроби.

friend fraction operator+(const fraction &fraction, double decimal)

Функция оператора сложения использует другой объект дроби, создаваемый из значения double, который добавляется к исходному объекту дроби.

friend fraction operator+(const fraction &fraction, int integer)

Функции оператора сложения использует другой объект дроби, создаваемый из значения int, который добавляется к исходному объекту дроби.

friend fraction operator+(double decimal, const fraction &fraction)

Функции оператора сложения использует исходный объект дроби, к которому добавляется новый объект дроби, создаваемый из значения double.

friend fraction operator+(int integer, const fraction &fraction)

Функции оператора сложения использует исходный объект дроби, к которому добавляется новый объект дроби, создаваемый из значения int.

fraction operator+=(const fraction &other)

Эта функция является методом класса fraction и используется для добавления другой дроби (представленной объектом other) к текущей дроби (представленной объектом this). Она возвращает ссылку на текущий объект fraction, поэтому может использоваться для цепочки операций.

Переменные, используемые в этой функции, включают:

* numerator и denominator, которые представляют числитель и знаменатель текущей дроби.
* integer, который представляет целую часть текущей дроби.
* sign, который определяет знак текущей дроби (true для отрицательных и false для положительных).
* numerator\_ и denominator\_, которые используются для временного хранения вычислений в процессе выполнения операции.
* other.numerator и other.denominator, которые представляют числитель и знаменатель дроби other.
* other.integer и other.sign, которые представляют целую часть и знак дроби other.

fraction operator+=(const int integer)

Оператор принимает целое число и создает временный объект fraction, который представляет это число в виде дроби. Затем он вызывает перегруженную версию оператора "+=" для объекта fraction с этим временным объектом и возвращает ссылку на текущий объект fraction.

fraction operator+=(const double decimal)

Оператор принимает число с плавающей точкой и создает временный объект fraction, который представляет это число в виде дроби. Затем он вызывает перегруженную версию оператора "+=" для объекта fraction с этим временным объектом и возвращает ссылку на текущий объект fraction.

**Текст программы**

**main.cpp**

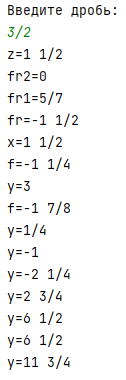
*/// Copyright 2023 ttsypyshev <ttsypyshev01@gmail.com>*#include <iostream>  
#include "../include/fraction.h"  
  
using namespace std;  
int main() {  
*//ввод дроби с клавиатуры* cout << "Введите дробь: \n";  
 fraction z;  
 cin >> z;  
 cout << "z=" << z << endl;  
*//проверка конструкторов* fraction fr1(10, 14), fr2;  
 cout << "fr2=" << fr2 << endl;  
 cout << "fr1=" << fr1 << endl;  
 fraction fr = "-1 4/8";  
 cout << "fr=" << fr << endl;  
 fraction x(z), y;  
 cout << "x=" << x << endl;  
 double dbl = -1.25;  
 fraction 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;  
 return 0;  
}

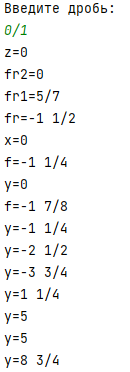
**class.h**

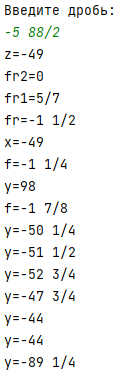
*/// Copyright 2023 ttsypyshev <ttsypyshev01@gmail.com>*#ifndef **INF\_LAB\_2\_FRACTION\_H**#define **INF\_LAB\_2\_FRACTION\_H**#include <iostream>  
#include <cstring>  
#include <cmath>  
  
int gcd(int a, int b) {  
 if (b == 0) return a;  
 return gcd(b, a % b);  
}  
  
class fraction {  
private:  
 bool sign;  
 unsigned integer, numerator, denominator;  
public:  
 fraction() {  
 sign = false;  
 integer = 0;  
 numerator = 0;  
 denominator = 1;  
 }  
  
 fraction(unsigned numerator\_, unsigned denominator\_, bool sign\_ = false, unsigned integer\_ = 0) {  
 sign = sign\_;  
 integer = integer\_ + numerator\_ / denominator\_;  
 int divisor = gcd(numerator\_, denominator\_);  
 numerator = numerator\_ / divisor;  
 denominator = denominator\_ / divisor;  
 }  
  
 fraction(const char \*input) {  
 if (input[0] == '-') {  
 sign = true;  
 ++input;  
 } else {  
 sign = false;  
 }  
  
 const char \*dot = strchr(input, ' ');  
 const char \*slash = strchr(input, '/');  
 if (dot != nullptr) {  
 integer = atoi(input);  
 input = dot + 1;  
 } else if (slash == nullptr) {  
 integer = atoi(input);  
 } else {  
 integer = 0;  
 }  
  
 if (slash != nullptr) {  
 unsigned numerator\_ = atoi(input), denominator\_ = atoi(slash + 1);  
 integer += numerator\_ / denominator\_;  
 int divisor = gcd(numerator\_, denominator\_);  
 numerator = numerator\_ / divisor;  
 denominator = denominator\_ / divisor;  
  
 } else {  
 numerator = 0;  
 denominator = 1;  
 }  
 }  
  
 fraction(const double decimal) {  
 sign = decimal < 0 ? true : false;  
 double abs\_num = std::abs(decimal);  
 int whole\_part = static\_cast<int>(abs\_num);  
 double fractional\_part = abs\_num - whole\_part;  
 int numerator\_ = static\_cast<int>(fractional\_part \* 1000000);  
 int denominator\_ = 1000000;  
 int divisor = gcd(numerator\_, denominator\_);  
 integer = whole\_part;  
 numerator = numerator\_ / divisor;  
 denominator = denominator\_ / divisor;  
 }  
  
 fraction(const int decimal) {  
 sign = decimal < 0 ? true : false;  
 integer = abs(decimal);  
 numerator = 0;  
 denominator = 1;  
 }  
  
 fraction(const fraction &item) {  
 sign = item.sign;  
 integer = item.integer;  
 numerator = item.numerator;  
 denominator = item.denominator;  
 }  
  
 friend std::istream &operator>>(std::istream &input, fraction &item) {  
 unsigned tmp = 0, integer\_ = 0, numerator\_ = 0, denominator\_ = 1;  
  
 if (input.peek() == '-') {  
 input.ignore();  
 item.sign = true;  
 } else {  
 item.sign = false;  
 }  
  
 input >> tmp;  
 if (input.peek() == '\n' || input.eof()) integer\_ = tmp;  
 else {  
 if (input.peek() == ' ') {  
 integer\_ = tmp;  
 input.ignore();  
 input >> tmp;  
 }  
 numerator\_ = tmp;  
 input.ignore();  
 input >> denominator\_;  
 }  
  
 int divisor = gcd(numerator\_ % denominator\_, denominator\_);  
 item.integer += integer\_ + numerator\_ / denominator\_;  
 item.numerator = (numerator\_ % denominator\_) / divisor;  
 item.denominator = denominator\_ / divisor;  
 return input;  
 }  
  
 friend std::ostream &operator<<(std::ostream &output, const fraction &item) {  
 if (item.sign && item.denominator != 0) output << '-';  
 if (item.integer != 0) output << item.integer << ' ';  
 if (item.numerator != 0) output << item.numerator << '/' << item.denominator;  
 if (item.integer == 0 && item.numerator == 0) output << '0';  
 return output;  
 }  
  
 fraction operator+(const fraction &other) const {  
 fraction result;  
 int numerator\_ = 0, denominator\_ = 0;  
 if (numerator == 0 && other.numerator == 0) {  
 result.integer = abs(integer \* (sign ? -1 : 1) + other.integer \* (other.sign ? -1 : 1));  
 result.sign = sign == other.sign ? false : true;  
 result.numerator = 0;  
 result.denominator = 1;  
 } else {  
 if (numerator == 0) {  
 numerator\_ = integer \* other.denominator \* (sign ? -1 : 1) +  
 (other.integer \* other.denominator + other.numerator) \* (other.sign ? -1 : 1);  
 denominator\_ = other.denominator;  
 result.sign = numerator\_ < 0 ? true : false;  
 result.integer = abs(numerator\_) / denominator\_;  
 } else if (other.numerator == 0) {  
 numerator\_ = (integer \* denominator + numerator) \* (sign ? -1 : 1) +  
 other.integer \* denominator \* (other.sign ? -1 : 1);  
 denominator\_ = denominator;  
 result.sign = numerator\_ < 0 ? true : false;  
 result.integer = abs(numerator\_) / denominator\_;  
 } else {  
 numerator\_ = (integer \* denominator + numerator) \* other.denominator \* (sign ? -1 : 1) +  
 (other.integer \* other.denominator + other.numerator) \* denominator \*  
 (other.sign ? -1 : 1);  
 denominator\_ = denominator \* other.denominator;  
 result.sign = numerator\_ < 0 ? true : false;  
 result.integer = abs(numerator\_) / denominator\_;  
 }  
 int divisor = gcd(abs(numerator\_) % denominator\_, denominator\_);  
 result.numerator = (abs(numerator\_) % denominator\_) / divisor;  
 result.denominator = denominator\_ / divisor;  
 }  
 return result;  
 }  
  
 friend fraction operator+(const fraction &fraction, double decimal) {  
 class fraction other(decimal);  
 return fraction + other;  
 }  
  
 friend fraction operator+(const fraction &fraction, int integer) {  
 class fraction other = integer;  
 return fraction + other;  
 }  
  
 friend fraction operator+(double decimal, const fraction &fraction) {  
 class fraction other = decimal;  
 return fraction + other;  
 }  
  
 friend fraction operator+(int integer, const fraction &fraction) {  
 class fraction other = integer;  
 return fraction + other;  
 }  
  
 fraction operator+=(const fraction &other) {  
 int numerator\_ = 0, denominator\_ = 0;  
 if (numerator == 0 && other.numerator == 0) {  
 integer = abs(integer \* (sign ? -1 : 1) + other.integer \* (other.sign ? -1 : 1));  
 sign = sign == other.sign ? false : true;  
 numerator = 0;  
 denominator = 1;  
 } else {  
 if (numerator == 0) {  
 numerator\_ = integer \* other.denominator \* (sign ? -1 : 1) +  
 (other.integer \* other.denominator + other.numerator) \* (other.sign ? -1 : 1);  
 denominator\_ = other.denominator;  
 sign = numerator\_ < 0 ? true : false;  
 integer = abs(numerator\_) / denominator\_;  
 } else if (other.numerator == 0) {  
 numerator\_ = (integer \* denominator + numerator) \* (sign ? -1 : 1) +  
 other.integer \* denominator \* (other.sign ? -1 : 1);  
 denominator\_ = denominator;  
 sign = numerator\_ < 0 ? true : false;  
 integer = abs(numerator\_) / denominator\_;  
 } else {  
 numerator\_ = (integer \* denominator + numerator) \* other.denominator \* (sign ? -1 : 1) +  
 (other.integer \* other.denominator + other.numerator) \* denominator \*  
 (other.sign ? -1 : 1);  
 denominator\_ = denominator \* other.denominator;  
 sign = numerator\_ < 0 ? true : false;  
 integer = abs(numerator\_) / denominator\_;  
 }  
 int divisor = gcd(abs(numerator\_) % denominator\_, denominator\_);  
 numerator = (abs(numerator\_) % denominator\_) / divisor;  
 denominator = denominator\_ / divisor;  
 }  
 return \*this;  
 }  
  
 fraction operator+=(const int integer) {  
 fraction other(integer);  
 \*this += other;  
 return \*this;  
 }  
  
 fraction operator+=(const double decimal) {  
 fraction other(decimal);  
 \*this += other;  
 return \*this;  
 }  
  
};  
  
#endif *//INF\_LAB\_2\_FRACTION\_H*

**Тестирование функций**

**Введём тестовые значения:**

****

****

****

**Вывод**

Мы успешно создали класс для работы с обыкновенными дробями, реализовали все операции с дробями, ввод и вывод дробей в строковом формате, конструкторы для инициализации объектов, а также операторы сложения и вывода для удобного контроля результатов программы.