Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Кафедра ПМиК

**Лабораторная работа № 5**

**по дисциплине «Современные технологии программирования»**

**«Абстрактный тип данных простая дробь»**

Выполнил: студент 4 курса ф. ИВТ, гр. ИП-813

Пещеров Вячеслав Александрович

Проверил: ассистент

к. ПМиК Агалаков А.А.

Новосибирск, 2021

Оглавление

[**Задание** 3](#_Toc90044825)

[**Реализация** 10](#_Toc90044826)

[**Демонстрация работы** 11](#_Toc90044827)

[**Вывод** 12](#_Toc90044828)

[**Список литературы** 13](#_Toc90044829)

[**Приложение** 14](#_Toc90044830)

[Листинг 1. TFrac.h 14](#_Toc90044831)

[Листинг 2. TFrac.cpp 15](#_Toc90044832)

# **Задание**

1. Реализовать абстрактный тип данных «простая дробь», используя класс

С++ в соответствии с приведенной ниже спецификацией.

2. Протестировать каждую операцию, определенную на типе данных,

используя средства модульного тестирования.

3. Если необходимо, предусмотрите возбуждение исключительных

ситуаций**.**

**Данные**

Простая дробь (тип TFrac) - это пара целых чисел: числитель и знаменатель (a/b). Простые дроби не изменяемые.

Операции

Операции могут вызываться только объектом простая дробь (тип **TFrac**), указатель на который в них передаётся по умолчанию. При описании операций этот объект называется «сама дробь» this.

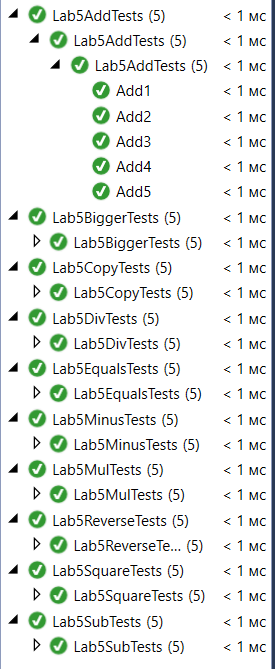
|  |  |
| --- | --- |
| ***Конструктор*** |  |
| Вход: | Пара целых чисел (a) и (b). |
| Предусловия: | b не равно 0. В противном случае  возбуждается исключение. |
| Процесс: | Инициализирует поля простой дроби (тип TFrac): числитель значением a, знаменатель - (b). В случае необходимости дробь предварительно сокращается.  Например:  *Конструктор*(6,3) = (2/1)  *Конструктор*(0,3) = (0/3). |
| Выход: | Нет. |
| Постусловия: | Поля объекта проинициализированы  начальными значениями. |
|  | |
| ***Конструктор*** |  |
| Вход: | Строковое представление простой дроби  . Например: ‘7/9’. |
| Предусловия: | b не равно 0. В противном случае  возбуждается исключение. |
| Процесс: | Инициализирует поля простой дроби (тип TFrac) строкой f =’a/b’. Числитель значением a, знаменатель - b. В случае необходимости дробь предварительно сокращается.  Например:  *Конструктор*(‘6/3’) = 2/1  *Конструктор* (‘0/3’) = 0/3 |
| Выход: | Нет. |
| Постусловия: | Поля объекта проинициализированы  начальными значениями. |
|  | |
| **Копировать:** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт копию самой дроби (тип TFrac) с числителем, и знаменателем такими же, как у самой дроби. |
| Выход: | Простая дробь (тип TFrac).  Например:  c = 2/1, Копировать(c) = 2/1 |
| Постусловия: | Нет. |
|  | |
| ***Сложить*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученную сложением самой дроби q = a1/b1 с d =a2/b2: ((a1/b1)+(a2/b2)=(a1\*b2 + a2\*b1)/( b1\* b2)).  Например: q = 1/2, d = -3/4  q.Сложить(d) = -1/4. |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
|  | |
| ***Умножить*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |
| Процесс: | Создаёт простую дробь (тип TFrac), полученную умножением самой дроби q  = a1/b1 на d = a2/b2 ((a1/b1)\*(a2/b2)=(a1\*  a2)/( b1\* b2)). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
|  | |
| ***Вычесть*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученную вычитанием d  = a2/b2 из самой дроби q = a1/b1: ((a1/b1)-(a2/b2)=(a1\* b2-a2\*b1)/(b1\*b2)).  Например: q = (1/2), d = (1/2)  q.Вычесть(d) = (0/1). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет |
|  | |
| ***Делить*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Числитель числа d не равно 0. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученное делением самой дроби q = a1/b1 на дробь d = a2/b2:  ((a1/b1)/(a2/b2)=(a1\* b2)/( a2\*b1)). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
|  | |
| ***Квадрат*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученную умножением самой дроби на себя: ((a/b)\*(a/b)=(a\* a)/(  b\* b)). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
|  | |
| ***Обратное*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает простую дробь  (тип TFrac), полученное делением единицы на саму дробь: 1/((a/b) = b/a. |
| Выход: | Простая дробь (тип TFrac) |
| Постусловия: | Нет. |
|  | |
| ***Минус*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт простую дробь, являющуюся разностью простых дробей z и q, где z - простая дробь (0/1), дробь, вызвавшая  метод. |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
|  | |
| ***Равно*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет |
| Процесс: | Сравнивает саму простую дробь q и d. Возвращает значение True, если q и d - тождественные простые дроби, и  значение False - в противном случае. |
| Выход: | Булевское значение. |
| Постусловия: | Нет. |
|  | |
| ***Больше*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |
| Процесс: | Сравнивает саму простую дробь q и d.  Возвращает значение True, если q > d, - значение False - в противном случае. |
| Выход: | Булевское значение. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьЧислительЧисло*** |  |
| Вход: |  |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение числителя дроби в  числовом формате. |
| Выход: | Вещественное значение. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьЗнаменательЧисло*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение знаменателя дроби  в числовом формате. |
| Выход: | Вещественное значение. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьЧислительСтрока*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение числителя дроби в  строковом формате. |
| Выход: | Строка. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьЗнаменательСтрока*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение знаменателя дроби  в строковом формате. |
| Выход: | Строка. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьДробьСтрока*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение простой дроби, в  строковом формате. |
| Выход: | Строка. |
| Постусловия: | Нет. |
|  |  |

***end TFrac***

# **Реализация**

В данной работе мы создали класс простой дроби и все необходимые для работы с ней операции, такие как: конструктор, копировать, сложить, умножить, вычесть, делить, квадрат, обратное, минус, равно, больше, взять число из числителя, взять число из знаменателя, взять числитель как строку, взять знаменатель как строку, взять дробь как строку.

# **Демонстрация работы**



# **Вывод**

Мы научились работать в среде Visual Studio, а именно разрабатывать в ней модульные тесты для тестирования наших функции и классов на языке C#. Данная среда отлично подходит для выполнения модульного тестирования и автоматизации.

# **Список литературы**

1. Подбельский В.В., Фомин С.С.инт Курс программирования на языке Си: учебник. – М.:ДМК Пресс, 2012 – 384 с.
2. Павловская Т.А. C#. Программирование на языке высокого уровня: Учебник для вузов. - СПб. : Питер, 2014 - 432 с. : ил. - (Серия "Учебник для вузов").
3. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4 на языке C# . 3-е изд.: - СПб.:Питер, 2012 - 928 с. : ил.
4. Котляров, В. П. Основы тестирования программного обеспечения : учебное пособие для СПО / В. П. Котляров. — Саратов : Профобразование, 2019 — 335 c. — ISBN 978-5-4488-0364-2. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/86202.html (дата обращения: 21.08.2020). — Режим доступа: для авторизир. пользователей

# **Приложение**

## Листинг 1. TFrac.h

#pragma once

#include <string>

#include <iostream>

using namespace std;

class TFrac

{

private:

int chisl;

int znam;

int gcd(int a, int b) const noexcept;

void fix() noexcept;

public:

class Zero : exception

{

public:

Zero(const char\* message) : exception(message)

{

}

};

explicit TFrac(int a, int b);

explicit TFrac(string a);

TFrac copy() const noexcept;

TFrac add(TFrac d) const noexcept;

TFrac sub(TFrac d) const noexcept;

TFrac mul(TFrac d) const noexcept;

TFrac div(TFrac d) const;

TFrac square() const noexcept;

TFrac reverse() const noexcept;

TFrac minus() const noexcept;

bool equals(TFrac d) const noexcept;

bool bigger(TFrac d) const noexcept;

int getchisl() const noexcept;

int getznam() const noexcept;

string str\_getprime() const noexcept;

string str\_getchisl() const noexcept;

string str\_getznam() const noexcept;

};

## Листинг 2. TFrac.cpp

#include "pch.h"

#include "TFrac.h"

int TFrac::gcd(int a, int b) const noexcept //проверка знаков

{

if (a == 0)

{

if (b < 0)

{

return -1 \* b;

}

return b;

}

if (a < 0)

{

a \*= -1;

}

if (b < 0)

{

b \*= -1;

}

while (a != b)

{

if (a > b)

{

a -= b;

}

else

{

b -= a;

}

}

return a;

}

void TFrac::fix() noexcept//перенос знака в числитель

{

if (this->znam < 0)

{

this->znam \*= -1;

this->chisl = -this->chisl;

}

}

int TFrac::getchisl() const noexcept//получаем числитель

{

return this->chisl;

}

int TFrac::getznam() const noexcept//получаем знаменатель

{

return this->znam;

}

string TFrac::str\_getchisl() const noexcept//получаем числитель как строку

{

return to\_string(this->chisl);

}

string TFrac::str\_getznam() const noexcept//получаем знаменатель

{

return to\_string(this->znam);

}

string TFrac::str\_getprime() const noexcept//получить дробь

{

return to\_string(this->chisl) + "/" + to\_string(this->znam);

}

TFrac::TFrac(int a, int b)//конструктор

{

if (b == 0)

{

throw Zero("ERROR 0 ZNAM");

}

this->chisl = (a / gcd(a, b));

this->znam = (b / gcd(a, b));

fix();

}

TFrac::TFrac(string tfrac)//конструктор для строки

{

int a = 0;

int b = 0;

string delimiter = "/";

string tokenA = tfrac.substr(0, tfrac.find(delimiter));

a = stoi(tokenA);

string tokenB = tfrac.substr(tfrac.find(delimiter) + delimiter.length(), std::string::npos);

b = stoi(tokenB);

if (b == 0)

{

throw Zero("ERROR 0 ZNAM");

}

this->chisl = (a / gcd(a, b));

this->znam = (b / gcd(a, b));

fix();

}

TFrac TFrac::copy() const noexcept// копируем

{

return TFrac(this->chisl, this->znam);

}

TFrac TFrac::add(TFrac d) const noexcept//добавить число

{

return TFrac

(

((this->chisl \* d.getznam() + this->znam \* d.getchisl()) / gcd(this->chisl, this->znam)),

((this->znam \* d.getznam()) / gcd(this->chisl, this->znam))

);

}

TFrac TFrac::sub(TFrac d) const noexcept//сложить дробь

{

return TFrac

(

((this->chisl \* d.getznam() - this->znam \* d.getchisl()) / gcd(this->chisl, this->znam)),

((this->znam \* d.getznam()) / gcd(this->chisl, this->znam))

);

}

TFrac TFrac::mul(TFrac d) const noexcept//перемножить дробь

{

return TFrac

(

((this->chisl \* d.getchisl()) / gcd(this->chisl, this->znam)),

((this->znam \* d.getznam()) / gcd(this->chisl, this->znam))

);

}

TFrac TFrac::div(TFrac d) const//поделить дробь

{

if (d.getchisl() == 0)

{

throw Zero("ERROR 0 CHISL IN DIV");

}

return TFrac

(

((this->chisl \* d.getznam()) / gcd(this->chisl, this->znam)),

((this->znam \* d.getchisl()) / gcd(this->chisl, this->znam))

);

}

TFrac TFrac::square() const noexcept//возвести в квадрат

{

return TFrac(this->chisl \* this->chisl, this->znam \* this->znam);

}

TFrac TFrac::reverse() const noexcept//ревёрс дроби

{

return TFrac(TFrac(1, 1).div(\*this));

}

TFrac TFrac::minus() const noexcept//сделать дробь отрицательной

{

return TFrac(TFrac(0, 1).sub(\*this));

}

bool TFrac::equals(TFrac d) const noexcept//сравнение дроби

{

if ((this->chisl \* d.getznam()) == (d.getchisl() \* this->znam))

{

return true;

}

else

{

return false;

}

}

bool TFrac::bigger(TFrac d) const noexcept//какая дробь больше

{

if ((this->chisl \* d.getznam()) > (d.getchisl() \* this->znam))

{

return true;

}

else

{

return false;

}

}