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 № 7**

**по дисциплине «Современные технологии программирования»**

**«Абстрактный тип данных p – ичное число»**

Выполнил: студент 4 курса ф.

ИВТ, гр. ИП-813

Пещеров Вячеслав Александрович

Проверил: ассистент

к. ПМиК Агалаков А.А.

Новосибирск, 2021

Оглавление

[**Задание** 3](#_Toc90045777)

[**Реализация** 12](#_Toc90045778)

[**Демонстрация работы** 13](#_Toc90045779)

[**Вывод** 14](#_Toc90045780)

[**Список литературы** 15](#_Toc90045781)

[**Приложение** 16](#_Toc90045782)

[Листинг 1. TPNumber.h 16](#_Toc90045783)

[Листинг 2. TPNumber.cpp 19](#_Toc90045784)

# **Задание**

1. Реализовать абстрактный тип данных «р-ичное число», используя класс, 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.

Спецификация типа данных «р-ичное число».

**ADT TPNumber**

**Данные**

Р-ичное число TPNumber - это действительное число (n) со знаком в системе счисления с основанием (b) (в диапазоне 2..16), содержащее целую и дробную части. Точность представления числа – (c >= 0). Р-ичные числа изменяемые.

**Операции**

Операции могут вызываться только объектом р-ичное число (тип TPNumber), указатель на который в них передаётся по умолчанию. При описании операций этот объект называется «само число».

|  |  |
| --- | --- |
| ***КонструкторЧисло*** |  |
| Вход: | Вещественное число (a). Система счисления (b), точность представления  числа (c) – целые числа. |
| Предусловия: | 2<=b<= 16, c >=0. В противном случае |

|  |  |
| --- | --- |
|  | возбуждается исключение. |
| Процесс: | Создаёт p-ичное число: система счисления (b), точность представления (c). В поле (n) созданного числа заносится (a).  Например:  NCreate(a,3,3) = число a в системе счисления 3 с тремя разрядами после троичной точки.  NCreate(a,3,2) = число a в системе счисления 3 с двумя разрядами после троичной точки. |
| Постусловия: | Поля объекта проинициализированы  начальными значениями. |
|  | |
| ***КонструкторСтрока*** |  |
| Вход: | Строковое представление р–ичного  числа (a), системы счисления (b), точности представления числа (c) |
| Предусловия: | 2<=b<= 16, c >=0. В противном случае  возбуждается исключение. |
| Процесс: | Создаёт р-ичное число: система счисления (b), точность представления (c). В поле (n) созданного числа заносится результат преобразования строки (a) в числовое представление. b-ичное число (a) и основание системы  счисления (b) представлены в формате |

|  |  |
| --- | --- |
|  | строки.  Например:  SCreate(‘20’,’3’,’6’) = 20 в  системе счисления 3, точность 6 знаков после запятой.  SCreate(‘0’,’3’,’8’) = 0 в системе счисления 3, точность 8 знаков после  запятой. |
| Постусловия: | Поля объекта проинициализированы  начальными значениями. |
|  | |
| **Копировать:** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт копию самого числа (тип TPNumber). |
| Выход: | р-ичное число. |
| Постусловия: | Нет. |
|  | |
| ***Сложить*** |  |
| Вход: | Р-ичное число d с основанием и  точностью такими же, как у самого числа. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает р-ичное число (тип TPNumber), полученное сложением полей (n) самого числа и  числа d. |

|  |  |
| --- | --- |
| Выход: | р-ичное число. |
| Постусловия: | Нет |
|  | |
| ***Умножить*** |  |
| Вход: | Р-ичное число d с основанием и точностью такими же, как у самого  числа. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает р-ичное число (тип TPNumber), полученное умножением полей (n) самого числа и  числа d. |
| Выход: | Р-ичное число (тип TPNumber). |
| Постусловия: | Нет. |
|  | |
| ***Вычесть*** |  |
| Вход: | Р-ичное число d с основанием и точностью такими же, как у самого  числа. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает р-ичное число (тип TPNumber), полученное вычитанием полей (n) самого числа и  числа d. |
| Выход: | Р-ичное число (тип TPNumber). |
| Постусловия: | Нет. |
|  | |
| ***Делить*** |  |
| Вход: | Р-ичное число d с основанием и |

|  |  |
| --- | --- |
|  | точностью такими же, как у самого  числа. |
| Предусловия: | Поле (n) числа (d) не равно 0. |
| Процесс: | Создаёт и возвращает р-ичное число (тип TPNumber), полученное делением полей (n) самого числа на поле (n)  числа d. |
| Выход: | Р-ичное число (тип TPNumber). |
| Постусловия: | Нет. |
|  | |
| ***Обратить*** |  |
| Вход: | Нет. |
| Предусловия: | Поле (n) самого числа не равно 0. |
| Процесс: | Создаёт р-ичное число, в поле (n) которого заносится значение,  полученное как 1/(n) самого числа. |
| Выход: | Р-ичное число (тип TPNumber). |
| Постусловия: | Нет. |
|  | |
| ***Квадрат*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт р-ичное число, в поле (n) которого заносится значение, полученное как квадрат поля (n)  самого числа. |
| Выход: | Р-ичное число (тип TPNumber). |
| Постусловия: | Нет. |
|  | |

|  |  |
| --- | --- |
| ***ВзятьРЧисло*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение поля (n) самого  числа. |
| Выход: | Вещественное значение. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьРСтрока*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает р-ичное число (q) в формате строки, изображающей значение поля (n) самого числа в  системе счисления (b) с точностью (c). |
| Выход: | Строка. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьОснованиеЧисло*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение поля (b) самого  числа (q). |
| Выход: | Целочисленное значение |
| Постусловия: | Нет. |
|  | |
| ***ВзятьОснованиеСтрока*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |

|  |  |
| --- | --- |
| Процесс: | Возвращает значение поля (b) самого числа в формате строки, изображающей (b) в десятичной  системе счисления. |
| Выход: | Строка. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьТочностьЧисло*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение поля (c) самого  числа . |
| Выход: | Целое значение. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьТочностьСтрока*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение поля (c) самого числа в формате строки, изображающей (c) в десятичной  системе счисления. |
| Выход: | Строка. |
| Постусловия: | Нет. |
|  | |
| ***УстановитьОснованиеЧисло*** |  |
| Вход: | Целое число (newb). |
| Предусловия: | 2 <= newb <= 16. |
| Процесс: | Устанавливает в поле (b) самого числа |

|  |  |
| --- | --- |
|  | значение (newb). |
| Выход: | Нет. |
| Постусловия: | Нет. |
|  | |
| ***УстановитьОснованиеСтрока*** |  |
| Вход: | Строка (bs), изображающая основание  (b) p-ичного числа в десятичной системе счисления. |
| Предусловия: | Допустимый диапазон числа,  изображаемого строкой (bs) - 2,,16. |
| Процесс: | Устанавливает значение поля (b) самого числа значением, полученным в результате преобразования строки  (bs). |
| Выход: | Строка. |
| Постусловия: | Нет. |
|  | |
| ***УстановитьТочностьЧисло*** |  |
| Вход: | Целое число (newc). |
| Предусловия: | newc >= 0. |
| Процесс: | Устанавливает в поле (c) самого числа  значение (newc). |
| Выход: | Нет. |
| Постусловия: | Нет. |
|  | |
| ***УстановитьТочностьСтрока*** |  |
| Вход: | Строка (newc). |
| Предусловия: | Строка (newc) изображает десятичное  целое >= 0. |

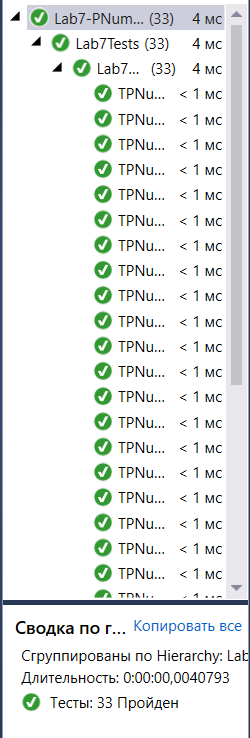
|  |  |
| --- | --- |
| Процесс: | Устанавливает в поле (c) самого числа  значение, полученное преобразованием строки (newc). |
| Выход: | Нет. |
| Постусловия: | Нет. |

*end TPNumber*

# **Реализация**

В данной работе мы создали класс p – ичных чисел и все необходимые для работы с ним операции, такие как: конструктор числа, конструктор строки, сложить, умножить, вычесть, делить, обратить, квадрат, взять число P, взять P как строку, взять основание числа, взять основание числа как строку, взять число с точностью, взять число с точностью как строку, установить в основание число, установить в основание заданную строку, установить число с определённой точностью, установить число с точностью 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. TPNumber.h

#pragma once

#include <string>

#include <iostream>

using namespace std;

class TPNumber

{

private:

double n;

int b;

int c;

public:

class constructor\_exception : exception

{

public:

explicit constructor\_exception(const char\* message) : exception(message)

{

}

};

class string\_constructor\_exception : exception

{

public:

explicit string\_constructor\_exception(const char\* message) : exception(message)

{

}

};

class add\_exception : exception

{

public:

explicit add\_exception(const char\* message) : exception(message)

{

}

};

class sub\_exception : exception

{

public:

explicit sub\_exception(const char\* message) : exception(message)

{

}

};

class mul\_exception : exception

{

public:

explicit mul\_exception(const char\* message) : exception(message)

{

}

};

class div\_exception : exception

{

public:

explicit div\_exception(const char\* message) : exception(message)

{

}

};

class reverse\_exception : exception

{

public:

explicit reverse\_exception(const char\* message) : exception(message)

{

}

};

class set\_b\_exception : exception

{

public:

explicit set\_b\_exception(const char\* message) : exception(message)

{

}

};

class set\_string\_b\_exception : exception

{

public:

explicit set\_string\_b\_exception(const char\* message) : exception(message)

{

}

};

class set\_c\_exception : exception

{

public:

explicit set\_c\_exception(const char\* message) : exception(message)

{

}

};

class set\_string\_c\_exception : exception

{

public:

explicit set\_string\_c\_exception(const char\* message) : exception(message)

{

}

};

explicit TPNumber(double n, int b, int c);

explicit TPNumber(const string& n, const string& b, const string& c);

TPNumber copy() const noexcept;

TPNumber add(const TPNumber& d) const;

TPNumber mul(const TPNumber& d) const;

TPNumber sub(const TPNumber& d) const;

TPNumber div(const TPNumber& d) const;

TPNumber reverse() const;

TPNumber square() const noexcept;

double get\_n() const noexcept;

string get\_n\_as\_string() const noexcept;

int get\_b() const noexcept;

string get\_b\_as\_string() const noexcept;

int get\_c() const noexcept;

string get\_c\_as\_string() const noexcept;

void set\_b(int newb);

void set\_b\_by\_string(const string& bs);

void set\_c(int newc);

void set\_c\_by\_string(const string& cs);

};

## Листинг 2. TPNumber.cpp

#include "pch.h"

#include "TPNumber.h"

TPNumber::TPNumber(double n, int b, int c)//конструктор числа

{

if (b <= 1 || b >= 17)

{

throw constructor\_exception("Bad constructor argument: base");

}

if (c < 0)

{

throw constructor\_exception("Bad constructor argument: precision");

}

this->n = n;

this->b = b;

this->c = c;

}

TPNumber::TPNumber(const string& n, const string& b, const string& c)// конструктор для string

{

if (stoi(b) <= 1 || stoi(b) >= 17)

{

throw string\_constructor\_exception("Bad string constructor argument: base");

}

if (stoi(c) < 0)

{

throw string\_constructor\_exception("Bad string constructor argument: precision");

}

this->n = stod(n);

this->b = stoi(b);

this->c = stoi(c);

}

TPNumber TPNumber::copy() const noexcept//копирование

{

return TPNumber(this->n, this->b, this->c);

}

TPNumber TPNumber::add(const TPNumber& d) const//добавление числа

{

if (d.b != this->b)

{

throw add\_exception("Incorrect add base");

}

if (d.c != this->c)

{

throw add\_exception("Incorrect add precision");

}

return TPNumber(this->n + d.n, this->b, this->c);

}

TPNumber TPNumber::mul(const TPNumber& d) const//перемножение

{

if (d.b != this->b)

{

throw mul\_exception("Incorrect mul base");

}

if (d.c != this->c)

{

throw mul\_exception("Incorrect mul precision");

}

return TPNumber(this->n \* d.n, this->b, this->c);

}

TPNumber TPNumber::sub(const TPNumber& d) const//вычитание

{

if (d.b != this->b)

{

throw sub\_exception("Incorrect sub base");

}

if (d.c != this->c)

{

throw sub\_exception("Incorrect sub precision");

}

return TPNumber(this->n - d.n, this->b, this->c);

}

TPNumber TPNumber::div(const TPNumber& d) const//деление

{

if (d.b != this->b)

{

throw div\_exception("Incorrect div base");

}

if (d.c != this->c)

{

throw div\_exception("Incorrect div precision");

}

if (d.n == 0)

{

throw div\_exception("Div: Division by zero");

}

return TPNumber(this->n / d.n, this->b, this->c);

}

TPNumber TPNumber::reverse() const//реверс числа

{

if (this->n == 0)

{

throw reverse\_exception("Reverse: Division by zero");

}

return TPNumber(1 / this->n, this->b, this->c);

}

TPNumber TPNumber::square() const noexcept//квадрат числа

{

return TPNumber(this->n \* this->n, this->b, this->c);

}

double TPNumber::get\_n() const noexcept//получить n

{

return this->n;

}

string TPNumber::get\_n\_as\_string() const noexcept//получить n как строку

{

string result;

int int\_part = (int)this->n;

double frac\_part = this->n - int\_part;

while (int\_part != 0)

{

auto digit = (int\_part % this->b);

if (digit >= 10)

{

result.insert(result.begin(), (char)(digit + 55));

}

else

{

result.insert(result.begin(), (char)(digit + 48));

}

int\_part /= this->b;

}

if (frac\_part != 0)

{

result += '.';

for (int exp = 0; exp < this->c; exp++)

{

auto digit = (char)(frac\_part \*= this->b);

if (digit >= 10)

{

result += (char)(digit + 55);

}

else

{

result += (char)(digit + 48);

}

frac\_part -= digit;

}

}

return result;

}

int TPNumber::get\_b() const noexcept//получить b

{

return this->b;

}

string TPNumber::get\_b\_as\_string() const noexcept//получить b как строку

{

return to\_string(this->b);

}

int TPNumber::get\_c() const noexcept//получить с

{

return this->c;

}

string TPNumber::get\_c\_as\_string() const noexcept//получить c как строку

{

return to\_string(this->c);

}

void TPNumber::set\_b(int newb)// установить b

{

if (newb <= 1 || newb >= 17)

{

throw set\_b\_exception("Incorrect newb");

}

this->b = newb;

}

void TPNumber::set\_b\_by\_string(const string& bs)//установить b как строку

{

if (stoi(bs) <= 1 || stoi(bs) >= 17)

{

throw set\_string\_b\_exception("Incorrect string newb");

}

this->b = stoi(bs);

}

void TPNumber::set\_c(int newc)//установить c

{

if (newc < 0)

{

throw set\_c\_exception("Incorrect newc");

}

this->c = newc;

}

void TPNumber::set\_c\_by\_string(const string& cs)//установить с как строку

{

if (stoi(cs) < 0)

{

throw set\_string\_c\_exception("Incorrect string newc");

}

this->c = stoi(cs);

}