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 № 8**

**по дисциплине «Современные технологии программирования»**

**«Редактор p – ичных чисел»**

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

Пещеров Вячеслав Александрович

Проверил: ассистент

к. ПМиК Агалаков А.А.

Новосибирск, 2021

Оглавление

[**Задание** 3](#_Toc90048460)

[**Реализация** 4](#_Toc90048461)

[**Демонстрация работы** 5](#_Toc90048462)

[**Вывод** 10](#_Toc90048463)

[**Список литературы** 11](#_Toc90048464)

[**Приложение** 12](#_Toc90048465)

[Листинг 1. Main.cpp 12](#_Toc90048466)

[Листинг 2. PEditor.h 13](#_Toc90048467)

[Листинг 3. PEditor.cpp 14](#_Toc90048468)

# **Задание**

1. Разработать и реализовать класс TEditor «Редактор р-ичных чисел»,

используя класс С++.

2. Протестировать каждую операцию, определенную на типе данных,

используя средства модульного тестирования по критерию С0.

3. Если необходимо, предусмотрите возбуждение исключительных

ситуаций.

# **Реализация**

В данной работе мы создали редактор p-ичных чисел и все необходимые для его работы операции, такие как: добавление числа, добавление нуля, изменение знака числа, удаление символа, добавления разделителя, очистить, считать строку, ввести строку, выйти и вывод промежуточного результата.

# **Демонстрация работы**

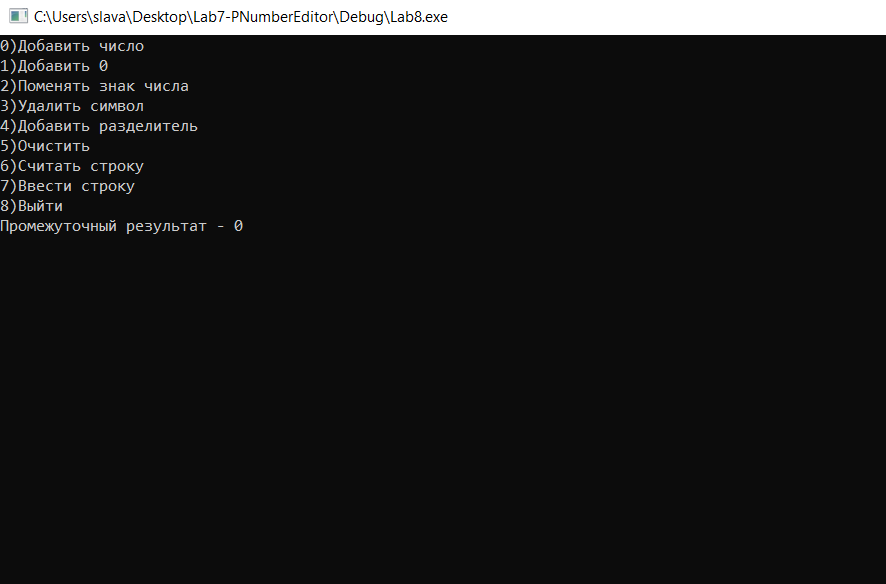


Рисунок 1. Меню программы

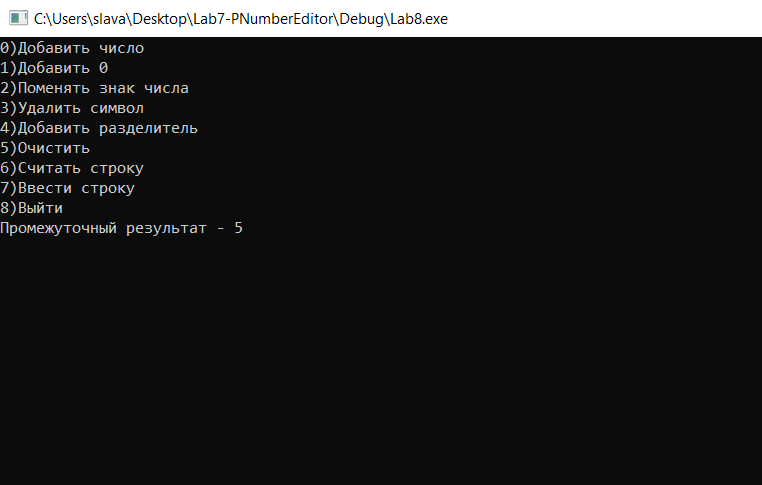


Рисунок 2. Добавление числа 5

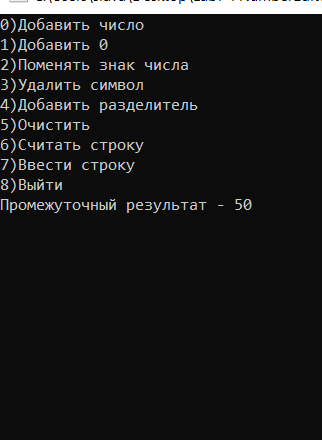


Рисунок 3. Добавление нуля

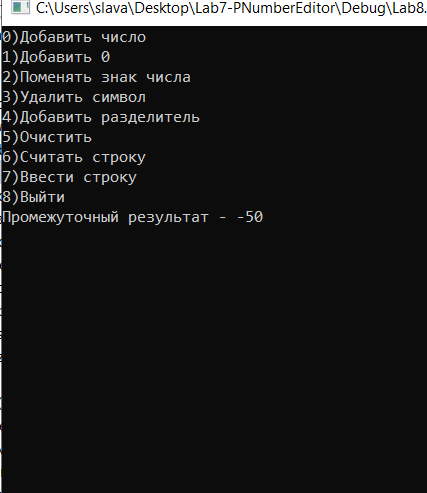


Рисунок 4. Изменили знак числа с – на +

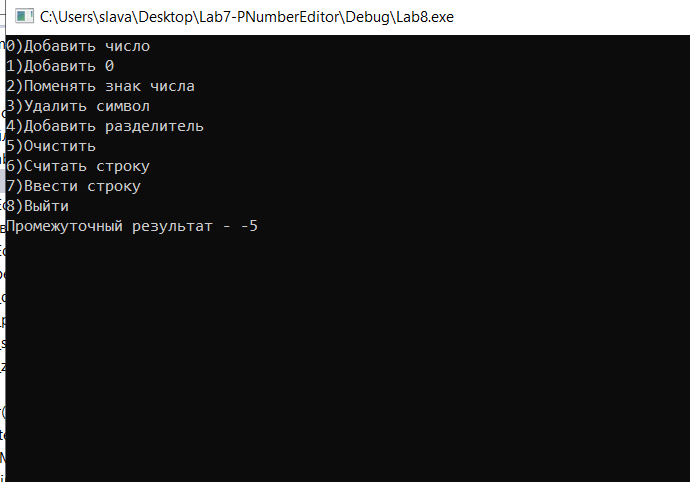


Рисунок 5. Удалили символ

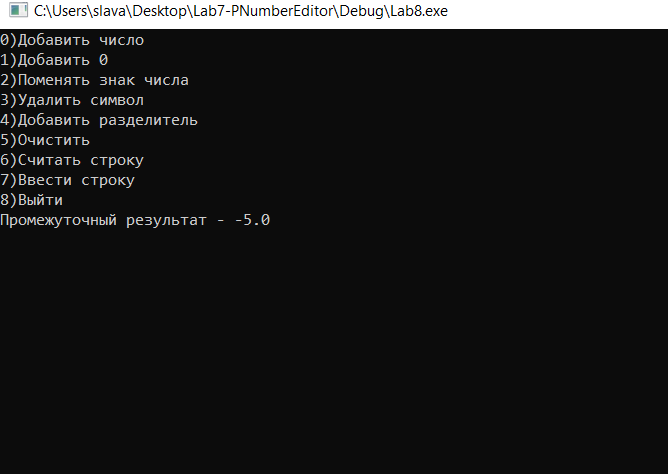


Рисунок 6. Добавили разделитель

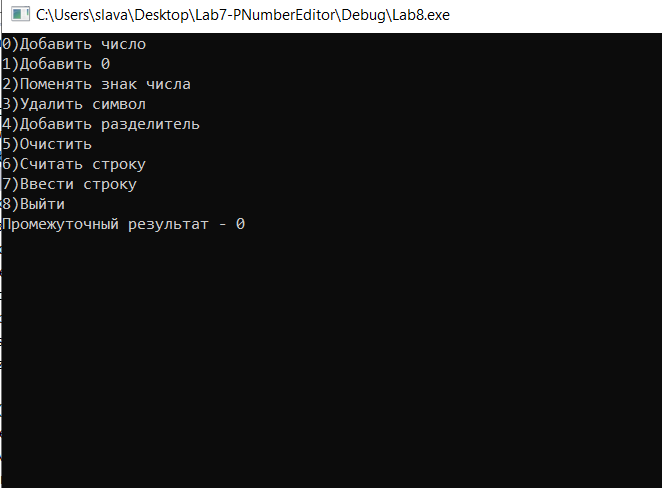
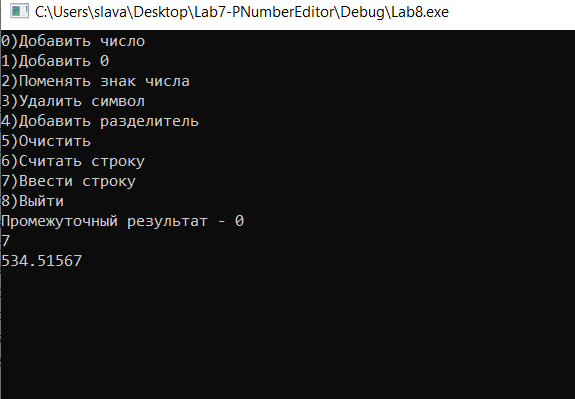


Рисунок 7. Очистили строку



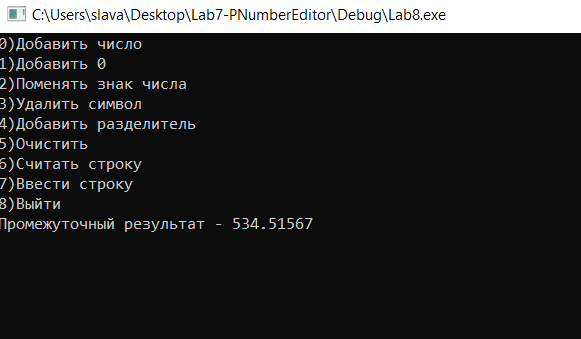


Рисунок 8, 9. Ввели строку с ручную

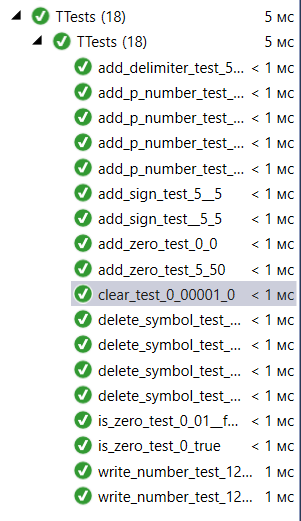


Рисунок 10. Тесты

# **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. Main.cpp

#include "pch.h"

#include "TPNumberEditor.h"

int main()

{

setlocale(LC\_ALL, "RUS");

int command = 0;

string res = "0";

TPNumberEditor a("0", 10, 1);

while (command != 8)

{

cout << "0)Добавить число" << endl;

cout << "1)Добавить 0" << endl;

cout << "2)Поменять знак числа" << endl;

cout << "3)Удалить символ" << endl;

cout << "4)Добавить разделитель" << endl;

cout << "5)Очистить" << endl;

cout << "6)Считать строку" << endl;

cout << "7)Ввести строку" << endl;

cout << "8)Выйти" << endl;

cout << "Промежуточный результат - " << res << endl;

cin >> command;

res = a.edit(command);

system("cls");

}

// std::cout << "Hello World!\n";

}

// Запуск программы: CTRL+F5 или меню "Отладка" > "Запуск без отладки"

// Отладка программы: F5 или меню "Отладка" > "Запустить отладку"

## Листинг 2. PEditor.h

#pragma once

#include <regex>

#include <string>

#include <iostream>

using namespace std;

class TPNumberEditor

{

typedef enum

{

ADD\_P\_NUMBER,

ADD\_ZERO,

ADD\_SIGN,

DELETE\_SYMBOL,

ADD\_DELIMITER,

CLEAR,

READ\_NUMBER,

WRITE\_NUMBER

}editor\_commands;

const char SIGN = '-';

const string ZERO = "0";

const string DELIMITER = ".";

private:

int base;

int precision;

string number;

public:

TPNumberEditor(const string& p\_number, int p\_base, int p\_precision);

bool is\_zero();

string add\_sign();

string add\_p\_number(int digit);

string add\_zero();

string delete\_symbol();

string add\_delimiter();

string clear();

string edit(int command);

string read\_number();

string write\_number(string a);

};

## Листинг 3. PEditor.cpp

#include "pch.h"

#include "TPNumberEditor.h"

TPNumberEditor::TPNumberEditor(const string &p\_number, int p\_base, int p\_precision)//конструктор

{

base = p\_base;

number = p\_number;

precision = p\_precision;

}

bool TPNumberEditor::is\_zero()//добавить 0

{

return number == ZERO || number == "-" + ZERO;

}

string TPNumberEditor::add\_sign()//добавить знак

{

if (number[0] == SIGN)

{

number.erase(number.begin());

}

else if (number != ZERO)

{

number = "-" + number;

}

return number;

}

string TPNumberEditor::add\_p\_number(int digit)//добавить число p

{

int delimiter\_index = number.find(DELIMITER);

char add\_symbol = (digit < 10 ? '0' + digit : 'A' + (digit - 10));

if (digit < base)

{

if (number[0] == '0' && delimiter\_index < 0)

{

number[0] = add\_symbol;

}

else if (number[0] == '-' && number[1] == '0')

{

number[1] = add\_symbol;

}

else if (delimiter\_index > 0)

{

string delimiter\_substr = number.substr(delimiter\_index);

if (delimiter\_substr.length() == 2 && number[number.length() - 1] == '0')

{

if (add\_symbol == '0')

{

number += add\_symbol;

}

else

{

number[number.length() - 1] = add\_symbol;

}

}

else

{

number += add\_symbol;

}

}

else

{

number += add\_symbol;

}

}

return number;

}

string TPNumberEditor::add\_zero()//добавить 0

{

return add\_p\_number(0);

}

string TPNumberEditor::delete\_symbol()//удалить символ

{

int delimiter\_index = number.find(DELIMITER);

if (delimiter\_index > 0)

{

string delimiter\_substr = number.substr(delimiter\_index);

if (delimiter\_substr.length() == 2)

{

number.pop\_back();

number.pop\_back();

}

else

{

number.pop\_back();

}

}

else

{

number.pop\_back();

}

if (number == "-" || number.empty())

{

number = ZERO;

}

return number;

}

string TPNumberEditor::clear()//очистить число

{

number = ZERO;

return number;

}

string TPNumberEditor::add\_delimiter()//добавить разделитель

{

if (number.find(DELIMITER) == std::string::npos)

{

number = number + DELIMITER + "0";

}

return number;

}

string TPNumberEditor::read\_number()

{

return number;

}

string TPNumberEditor::write\_number(string a)//вывести число

{

int digit = 0;

regex regex("-?([0-9A-F]+),?[0-9A-F]\*");

if (regex\_match(a, regex))

{

for (int i = 0; i < a.length(); i++)

{

switch (a[i])

{

case '0':

digit = 0;

break;

case '1':

digit = 1;

break;

case '2':

digit = 2;

break;

case '3':

digit = 3;

break;

case '4':

digit = 4;

break;

case '5':

digit = 5;

break;

case '6':

digit = 6;

break;

case '7':

digit = 7;

break;

case '8':

digit = 8;

break;

case '9':

digit = 9;

break;

case 'A':

digit = 10;

break;

case 'B':

digit = 11;

break;

case 'C':

digit = 12;

break;

case 'D':

digit = 13;

break;

case 'E':

digit = 14;

break;

case 'F':

digit = 15;

break;

case '.':

digit = 0;

break;

}

if (digit < base)

{

continue;

}

else

{

return number;

}

}

}

number = a;

return number;

}

string TPNumberEditor::edit(int command)//изменить чсило

{

string p;

int digit;

string res;

switch (command)

{

case ADD\_P\_NUMBER:

cin >> digit;

res = add\_p\_number(digit);

break;

case ADD\_ZERO:

res = add\_zero();

break;

case ADD\_SIGN:

res = add\_sign();

break;

case DELETE\_SYMBOL:

res = delete\_symbol();

break;

case ADD\_DELIMITER:

res = add\_delimiter();

break;

case CLEAR:

res = clear();

break;

case READ\_NUMBER:

res = read\_number();

break;

case WRITE\_NUMBER:

cin >> p;

res = write\_number(p);

break;

default:

res = number;

break;

}

return res;

}