

Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет
телекоммуникаций и информатики»

Лабораторная работа №4

Выполнил: студент 4 курса

ИВТ, гр. ИП-713

Михеев Н.А.

Проверил: ассистент кафедры

ПМиК

Агалаков А.А.

Новосибирск, 2020 г.

Цель

Сформировать практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов C++ и их модульного тестирования.

Задание

1. Разработать и реализовать класс TEditor «Редактор r-ичных чисел», используя класс C++.
2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования.
3. Если необходимо, предусмотрите возбуждение исключительных ситуаций.

Реализация и описание

Абстрактный тип данных «Редактор r-ичных чисел» был реализован посредством создания класса. Класс имеет поле, который хранит в себе число в типе string. Также были реализованы процедуры для взаимодействия с объектами данного класса.

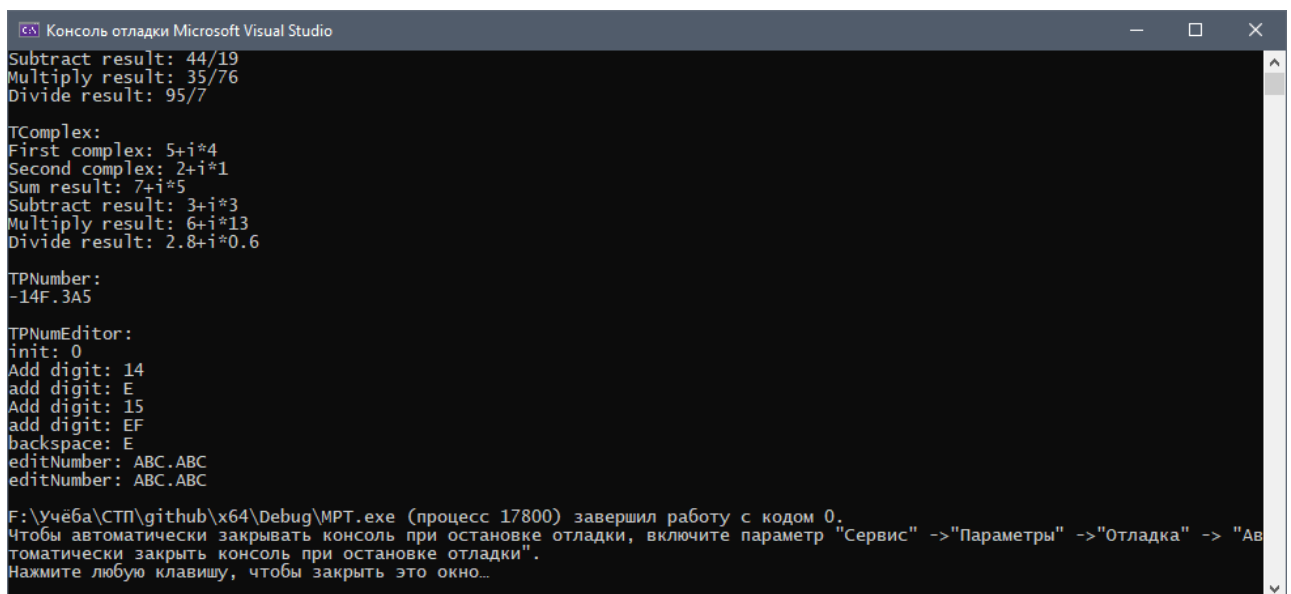
- TPNumEditor::TPNumEditor()– конструктор с инициализацией числа 0.
- TPNumEditor::equalsZero()– метод для проверки объекта на равенство “0”.
- TPNumEditor::changeSign() – метод для изменения знака числа.
- TPNumEditor::addDigit(int digit) – метод для добавления цифры (0-F) в число, хранящееся в переменной объекта.
- TPNumEditor::addZero() – метод для добавления нуля в число.
- TPNumEditor::backspace() – метод, удаляющий крайний правый символ из строки с числом.

- `TPNumEditor::clear()` – метод, очищающий строку и устанавливающий “0”.
- `TPNumEditor::addDivider()` – метод, устанавливающий знак деления между целой и дробной частями числа.
- `TPNumEditor::getNumberString()` – метод для получения значения числа в формате string.
- `TPNumEditor::setNumber(int num)` – метод для изменения числа.
- `TPNumEditor::editNumber(string input)` – метод для изменения числа объекта с помощью string.
- `TPNumEditor::menu(action action)` – метод, реализующий меню для взаимодействия с классом.

Заключение

В ходе данной работы согласно спецификациям задания был реализован абстрактный тип данных «р-ичное число». Также был получен практический опыт написания модульных тестов для тестирования каждой операции.

Скриншоты



```

Консоль отладки Microsoft Visual Studio
Subtract result: 44/19
Multiply result: 35/76
Divide result: 95/7

TComplex:
First complex: 5+i*4
Second complex: 2+i*1
Sum result: 7+i*5
Subtract result: 3+i*3
Multiply result: 6+i*13
Divide result: 2.8+i*0.6

TPNumber:
-14F.3A5

TPNumEditor:
init: 0
Add digit: 14
add digit: E
Add digit: 15
add digit: EF
backspace: E
editNumber: ABC.ABC
editNumber: ABC.ABC

F:\Учеба\СТП\github\х64\Debug\MPT.exe (процесс 17800) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрывать консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...

```

Рис. 1 – пример работы программы

Обозреватель тестов			
<div> <div>▶▶▶</div> <div>116</div> <div>116</div> <div>0</div> <div>Поиск в обозревателе тестов</div> </div>			
Тестирование	Длительность	Признаки	Сообщение об ошибке
<div> <div>✓</div> <div>MPT_Test (116)</div> </div>	< 1 мс		
<div> <div> <div>✓</div> <div>MPT_Tests (116)</div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>TComplexTest (35)</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>TFracTest (35)</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>TPNumEditorTest (17)</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>ADD_DIGIT_1</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>ADD_DIGIT_2</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>ADD_DIVIDER_1</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>ADD_ZERO_1</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>CHANGE_SIGN_1</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>CHANGE_SIGN_2</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>CLEAR_1</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>CONSTR_1</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>EDIT_NUMBER_1</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>EDIT_NUMBER_2</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>EDIT_NUMBER_3</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>EQ_ZERO_1</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>EQ_ZERO_2</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>GETNUMBERSTRING_1</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>SET_NUMBER_1</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>SET_NUMBER_2</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>SET_NUMBER_3</div> </div> </div> </div>	< 1 мс		
<div> <div> <div> <div>✓</div> <div>TPNumberTest (29)</div> </div> </div> </div>	< 1 мс		

Рис. 2 – сводка проведённого тестирования программы

Код программы

TPNumEditor.h

```

#pragma once
#include <string>
#include <regex>
#include <iostream>

using namespace std;

enum action
{
    _changeSign,
    _addDigit,
    _addZero,
    _backspace,
    _clear,
    _addDivider,
    _editNumber,
};

class TPNumEditor

```

```

{
private:
    std::string str;
public:
    TPNuEditor();
    bool equalsZero();
    string changeSign();
    string addDigit(int digit);
    string addZero();
    string backspace();
    string clear();
    string addDivider();
    string getNumberString();
    void setNumber(int num);
    string editNumber(string str);
    string menu(action action);
};

```

TPNuEditor.cpp

```

#include "TPNuEditor.h"

TPNuEditor::TPNuEditor()
{
    this->str = string("0");
}

bool TPNuEditor::equalsZero()
{
    return this->str == "0";
}

string TPNuEditor::changeSign()
{
    if (this->str[0] == '-')
        this->str.erase(0, 1);
    else if (this->str != "0")
        this->str.insert(0, "-");
    return this->str;
}

string TPNuEditor::addDigit(int digit)
{
    if (digit < 0 || digit > 16)
        return this->str;

    string symbols = "0123456789ABCDEF";
    if (equalsZero())
        return this->str = symbols[digit];
    this->str.push_back(symbols[digit]);
    return this->str;
}

string TPNuEditor::addZero()
{
    return this->addDigit(0);
}

string TPNuEditor::backspace()
{
    this->str.pop_back();
    if (this->str.empty() || this->str == "-")
        return this->addZero();
    if (this->str[str.size() - 1] == '.')
        this->str.pop_back();
    return this->str;
}

string TPNuEditor::clear()
{
    return this->str = "0";
}

string TPNuEditor::addDivider()
{
    if (this->str.find(".") == string::npos)
        this->str.append(".");
}

```

```

        return this->str;
    }

    string TPNuEditor::getNumberString()
    {
        return this->str;
    }

    void TPNuEditor::setNumber(int num)
    {
        this->str = to_string(num);
    }

    string TPNuEditor::editNumber(string input)
    {
        regex symbols("-?(0|[1-9A-F][0-9A-F]*).[0-9A-F]*");
        if (regex_match(input, symbols))
            return this->str = input;
        return this->str;
    }

    string TPNuEditor::menu(action action)
    {
        string input;
        int num;
        switch (action)
        {
            case _addDigit:
                cout << "Add digit: ";
                cin >> num;
                return this->addDigit(num);
                break;
            case _addZero:
                return this->addZero();
                break;
            case _backspace:
                return this->backspace();
                break;
            case _clear:
                return this->clear();
                break;
            case _addDivider:
                return this->addDivider();
                break;
            case _editNumber:
                cout << "editNumber: ";
                cin >> input;
                return this->editNumber(input);
                break;
            default:
                return this->str;
                break;
        }
        return this->str;
    }
}

```

TPNuEditor_Test.cpp

```

#include "pch.h"
#include "CppUnitTest.h"

#include "../MPT/TPNuEditor.h"
#include "../MPT/TPNuEditor.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace MPT_Tests
{
    TEST_CLASS(TPNuEditorTest)
    {
    public:
        TEST_METHOD(CONSTR_1)
        {
            Assert::AreEqual(string("0"), TPNuEditor().getNumberString());
        }

        TEST_METHOD(EQ_ZERO_1)

```

```

{
    Assert::AreEqual(true, TPNumEditor().equalsZero());
}

TEST_METHOD(EQ_ZERO_2)
{
    TPNumEditor tpneditor;
    tpneditor.addDigit(5);
    Assert::AreEqual(false, tpneditor.equalsZero());
}

TEST_METHOD(CHANGE_SIGN_1)
{
    TPNumEditor o1;
    o1.addDigit(5);
    Assert::AreEqual(string("-5"), o1.changeSign());
}

TEST_METHOD(CHANGE_SIGN_2)
{
    TPNumEditor o1;
    o1.addDigit(15);
    Assert::AreEqual(string("-F"), o1.changeSign());
}

TEST_METHOD(ADD_DIGIT_1)
{
    TPNumEditor o1;
    o1.addDigit(5);
    Assert::AreEqual(string("5"), o1.getNumberString());
}

TEST_METHOD(ADD_DIGIT_2)
{
    TPNumEditor o1;
    o1.addDigit(2);
    Assert::AreEqual(string("2"), o1.getNumberString());
}

TEST_METHOD(ADD_ZERO_1)
{
    TPNumEditor o1;
    o1.addDigit(2);
    o1.addZero();
    Assert::AreEqual(string("20"), o1.getNumberString());
}

TEST_METHOD(CLEAR_1)
{
    TPNumEditor o1;
    o1.addDigit(2);
    o1.clear();
    Assert::AreEqual(string("0"), o1.getNumberString());
}

TEST_METHOD(GETNUMBERSTRING_1)
{
    TPNumEditor o1;
    Assert::AreEqual(string("0"), o1.getNumberString());
}

TEST_METHOD(ADD_DIVIDER_1)
{
    TPNumEditor o1;
    o1.addDigit(5);
    o1.addDivider();
    o1.addDigit(5);
    Assert::AreEqual(string("5.5"), o1.getNumberString());
}

TEST_METHOD(SET_NUMBER_1)
{
    TPNumEditor o1;
    o1.setNumber(111);
    Assert::AreEqual(string("111"), o1.getNumberString());
}

```

```

TEST_METHOD(SET_NUMBER_2)
{
    TPNumericUpDown o1;
    o1.SetNumber(222);
    Assert::AreEqual(string("222"), o1.GetNumberString());
}

TEST_METHOD(SET_NUMBER_3)
{
    TPNumericUpDown o1;
    o1.SetNumber(333);
    Assert::AreEqual(string("333"), o1.GetNumberString());
}

TEST_METHOD(EDIT_NUMBER_1)
{
    TPNumericUpDown o1;
    o1.EditNumber("123");
    Assert::AreEqual(string("123"), o1.GetNumberString());
}

TEST_METHOD(EDIT_NUMBER_2)
{
    TPNumericUpDown o1;
    o1.EditNumber("321.321");
    Assert::AreEqual(string("321.321"), o1.GetNumberString());
}

TEST_METHOD(EDIT_NUMBER_3)
{
    TPNumericUpDown o1;
    o1.EditNumber("10");
    Assert::AreEqual(string("10"), o1.GetNumberString());
}
};

}

```