Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Сибирский государственный университет
телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Современные технологии программирования

Практическая работа №6

«Редактор комплексных чисел»

Выполнил: студент 4 курса
группы ИП-111
Кузьменок Денис Витальевич

Проверил преподаватель:
Зайцев Михаил Георгиевич

Новосибирск, 2024 г.

Цель:

Сформировать практические навыки реализации классов средствами объектно-ориентированного программирования C++.

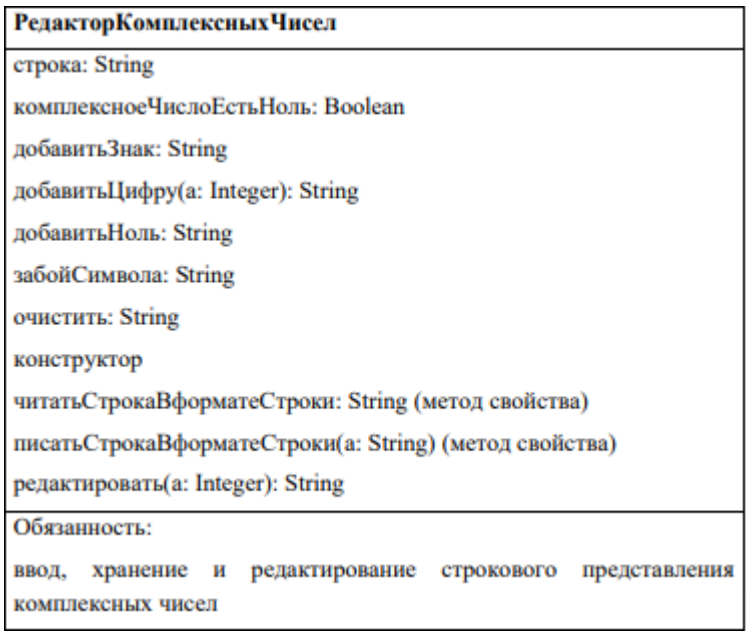
Задание:

1. Разработать и реализовать класс «Ввод и редактирование комплексных чисел» (TEditor), используя класс C++.
2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования Visual Studio по критерию C2.
3. Если необходимо, предусмотрите возбуждение исключительных ситуаци.
4. Класс должен отвечать за посимвольный ввод, хранение и редактирование строкового представления комплексных чисел. Значение комплексного нуля - ‘0, i* 0,’.

   Класс должен обеспечивать:

   - добавление цифры;
   - добавление и изменение знака действительной и мнимой частей;
   - добавление разделителя целой и дробной частей действительной и мнимой частей комплексного числа;
   - добавление разделителя мнимой и действительной частей комплексного числа
   - забой символа, стоящего справа (BackSpace);
   - установку нулевого значения комплексного числа (Clear);
   - чтение строкового представления комплексного числа;
   - запись строкового представления комплексного числа.

На Унифицированном языке моделирования UML (Unified Modeling Language) наш класс можно обозначить следующим образом:



Рекомендации к выполнению:

1. В классе TEditor опишите следующие атрибуты: • «строка» - строкового типа, содержит строковое представление редактируемого комплексного числа, .
2. В классе опишите следующие операции:

   • «число есть ноль», операция возвращает булевское значение True, если «строка» содержит изображение комплексного числа равного 0, +i 0,, False – в противном случае;

   • «добавить знак», операция добавляет или удаляет знак «-» из «строка» и возвращает значение «строка»;

   • «добавить цифру», операция получает целое число (числовое обозначение арабской цифры), преобразует его в символ и добавляет к «строка», если это допускает формат, возвращает значение «строка»;

   • «добавить ноль», операция добавляет ноль к «строка», если это допускает формат, возвращает значение «строка»;

- «забой символа», операция удаляет крайний правый символ «строка» и возвращает значение «строка»;
- «очистить», операция устанавливает в «строка» строку, изображающую комплексное число 0, +i 0,, возвращает значение «строка»;
- «редактировать», операция получает номер команды редактирования, выполняет действия по её выполнению и возвращает значение «строка»;
- «конструктор», создаёт объект типа TEditor;
- «читать «строка» в формате строки» - строкового типа (метод свойства), возвращает значение «строка» в заданном пользователем формате;
- «писать «строка» в формате строки», получает значение строкового типа (метод свойства) и заносит его в «строка»;

3. Класс реализуйте в отдельном модуле UEditor. В разделе описания констант опишите следующие константы: • «разделитель целой и дробной частей действительной и мнимой частей комплексного числа» - строкового типа; • «разделитель действительной и мнимой частей комплексного числа» - строкового типа; • «строковое представление нуля» - строкового типа.

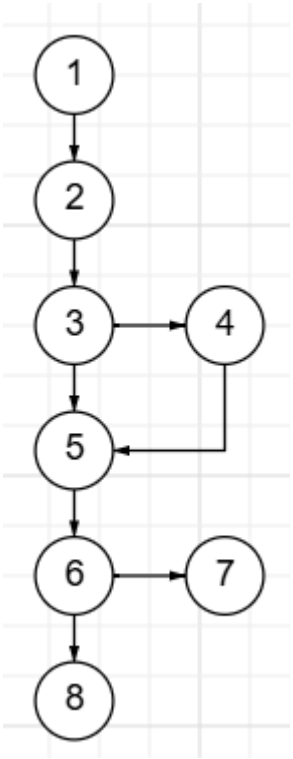
Реализация:

```
Удаление минуса в действительной части num1: 1,36+(i*3,08)
Удаление символа в действительной части num1: 0,36+(i*3,08)
Добавлние цифры в действительную часть num1: 8,36+(i*3,08)
Добавлние ещё одной цфиры в действительную часть num1: 83,36+(i*3,08)
Удаление цифры из действительной дробной части num1: 83,3+(i*3,08)
Измнение знака мнимой части на противоположный num1: 83,3-(i*3,08)
Изменение на исходный знак мнимой части num1: 83,3+(i*3,08)
Удаление из мнимой части цифры в целой части num1: 83,3+(i*0,08)
Добавление цифры в мнимую дробную часть num1: 83,3+(i*0,084)
Добавление ещё одной цифры в мнимую дробную часть num1: 83,3+(i*0,0845)

Исходное число num2: 43,81-(i*2,5)
num2 нулевое: нет
Измнение знака мнимой части num2: 43,81+(i*2,5)
Очищенное значение num2: 0,+(i*0,)
num2 нулевое: да

Введите число:
0 - Изменить знак на противоположный
1 - Добавить цифру
2 - Добавить ноль
3 - Удалить цифру
4 - Очистить комплексное число
5 - Записать комплексное число
6 - Изменить режим работы между действительной и мнимой частью
7 - Изменить режим работы между целой и дробной частью
8 - Показать комплексное число

Нажмите на клавишу Esc для завершения.
```

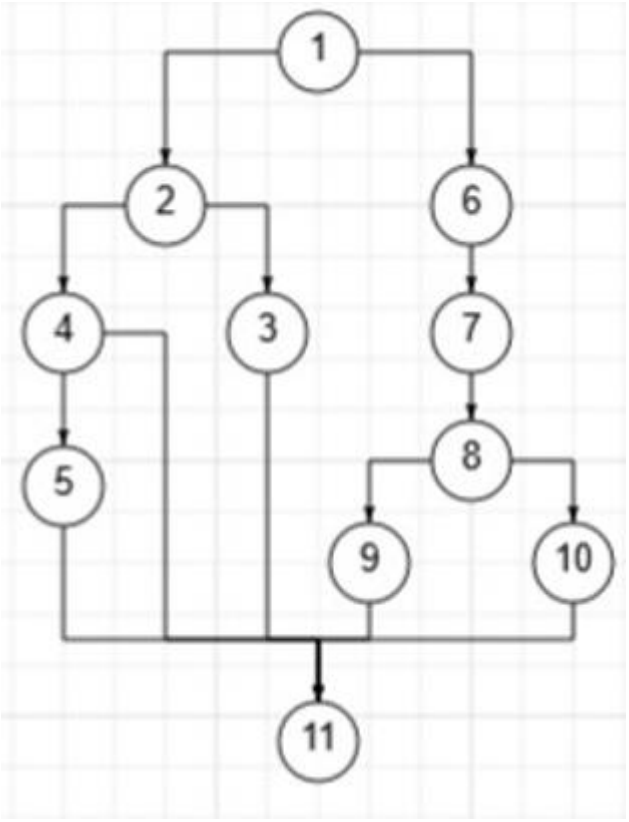
Рис. 1 – Результат проверки работоспособности программы.

По готовым функциям, были построены управляющие графы программы:

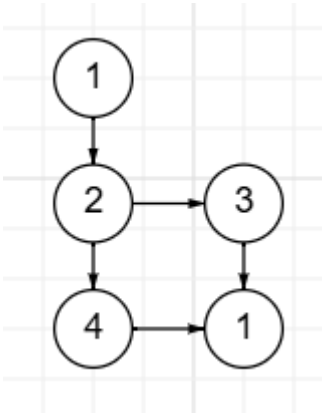
```
public bool IsZero():
```



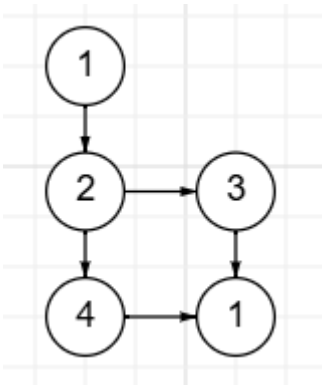

```
public string ToggleMinus():
```



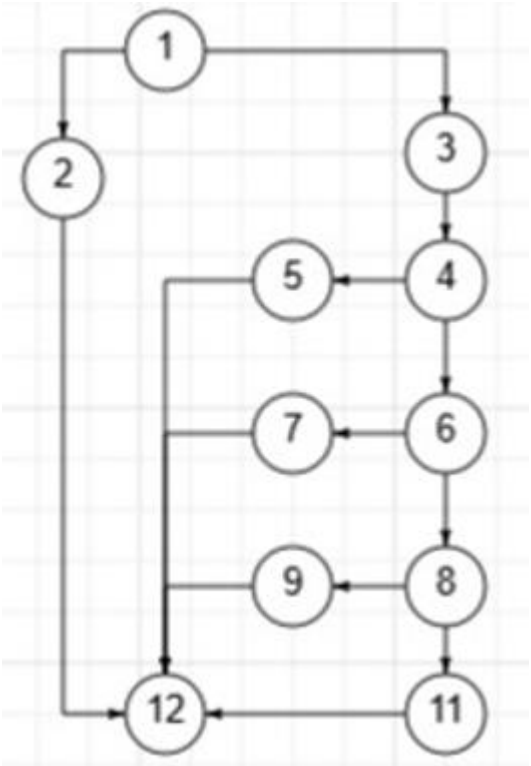

```
public PartToEdit ToggleMode():
```




```
public NumberPartToEdit ToggleNumberMode():
```

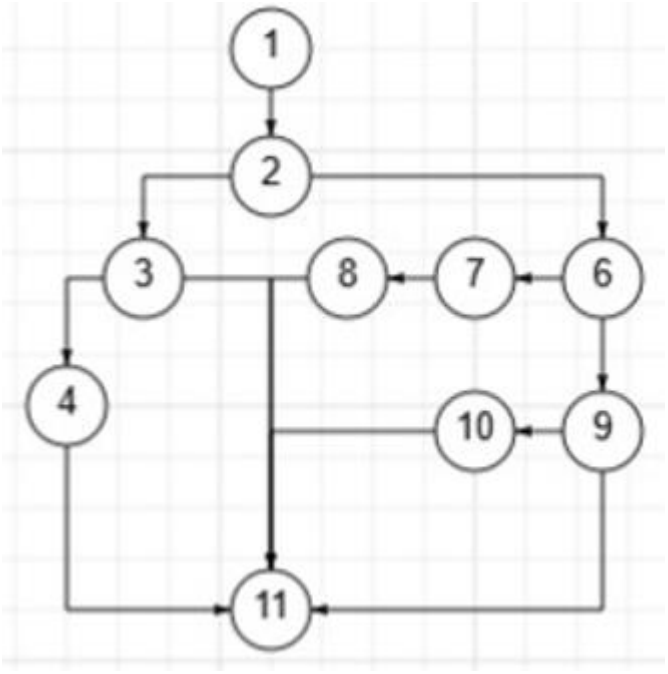



```
public string AddNumber(int a):
```

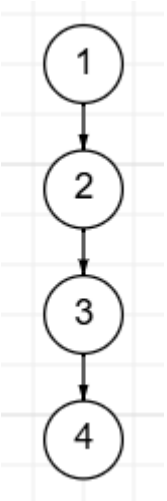



```
public string AddZero():
```

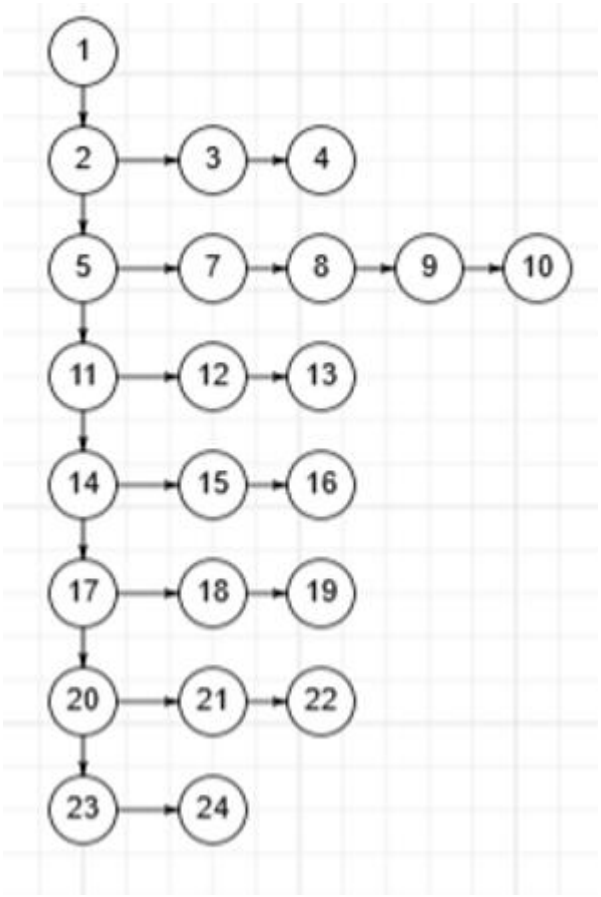

```
public string DelNumber():
```



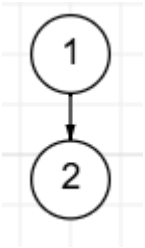

```
public string Clear():
```




```
public string Edit(int command):
```




```
public string WriteNumber(string otherNumber):
```




```
public string ReadNumber():
```

Описание тестовых данных

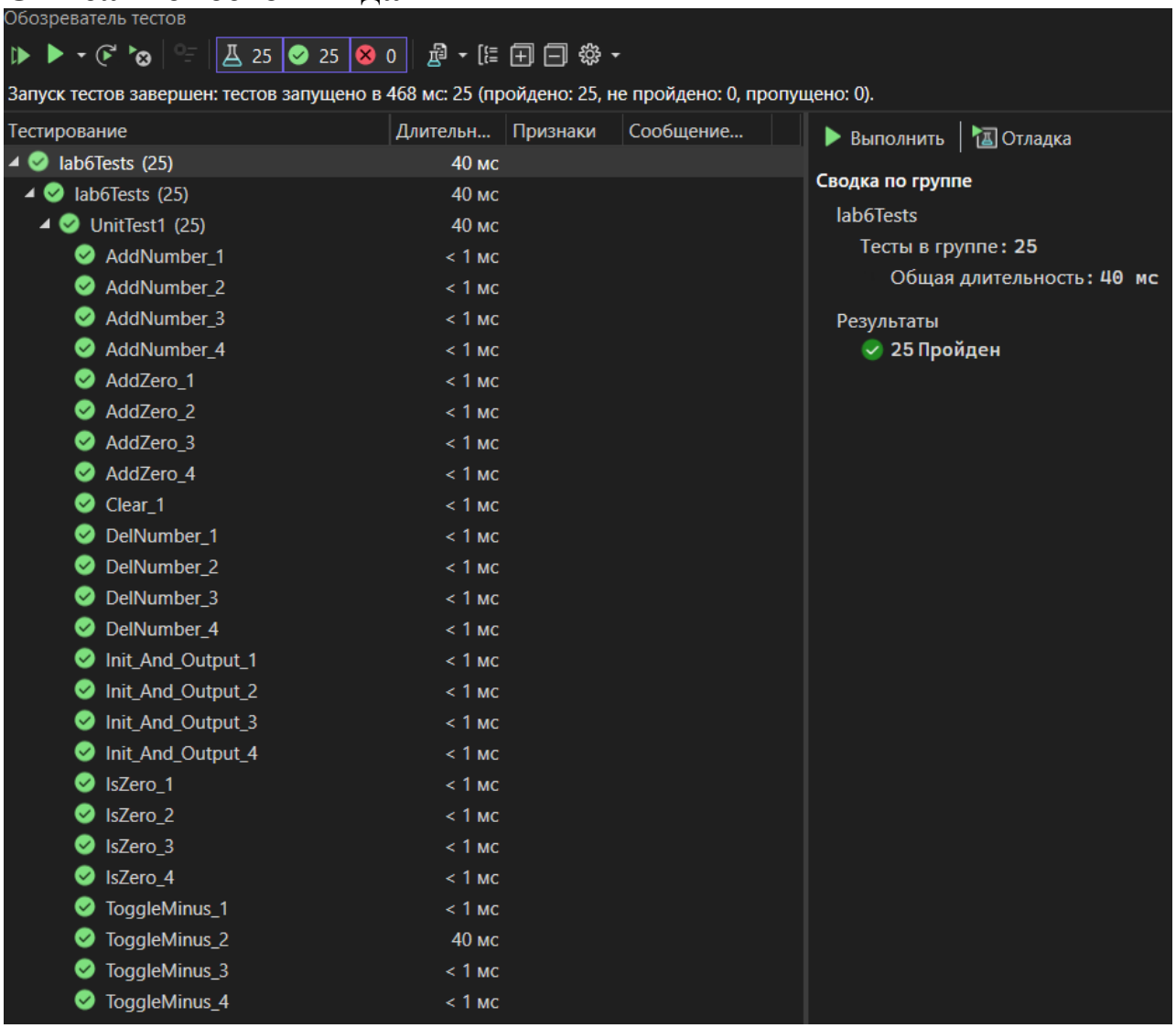


Рис. 2 – Результат выполнения модульных тестов.

1. Init_And_Output_1

- **Метод**: TEditor()
- **Входные данные**: Новый объект TEditor с параметрами вида 10,3+(i*0,8).
- **Ожидаемые данные**: 10,3+(i*0,8).

2. Init_And_Output_2

- **Метод**: TEditor()
- **Входные данные**: Новый объект TEditor с параметрами вида -12,6-(i*66,2).
- **Ожидаемые данные**: -12,6-(i*66,2).

3. Init_And_Output_3

- **Метод**: TEditor()
- **Входные данные**: Новый объект TEditor с параметрами вида 0,3+(i*0,0).
- **Ожидаемые данные**: 0,3+(i*0,0).

4. Init_And_Output_4

- **Метод**: TEditor()
- **Входные данные**: Пустой объект класса TEditor.
- **Ожидаемые данные**: 0,+(i*0,)

5. IsZero_1

- **Метод**: IsZero()
- **Входные данные**: Объект класса TEditor вида 12,36+(i*12,35).
- **Ожидаемые данные**: false.

6. IsZero_2

- **Метод**: IsZero()
- **Входные данные**: Объект класса TEditor вида 0,+(i*0,).
- **Ожидаемые данные**: true.

7. IsZero_3

- **Метод**: IsZero()
- **Входные данные**: Объект класса TEditor вида 0,+(i*12,54).
- **Ожидаемые данные**: false.

8. IsZero_4

- **Метод**: IsZero()
- **Входные данные**: Объект класса TEditor вида 0,43+(i*0,).
- **Ожидаемые данные**: false.

9. ToggleMinus_1

- **Метод**: ToggleMinus()
- **Входные данные**: Объект класса TEditor вида 12,36+(i*12,35).
- **Ожидаемые данные**: -12,36+(i*12,35).

10. ToggleMinus_2

- **Метод**: ToggleMinus()
- **Входные данные**: Объект класса TEditor вида -12,36+(i*12,35).
- **Ожидаемые данные**: 12,36+(i*12,35).

11. ToggleMinus_3

- **Метод**: ToggleMinus()
- **Входные данные**: Объект класса TEditor вида 12,36+(i*12,35).
- **Ожидаемые данные**: 12,36-(i*12,35).

12. ToggleMinus_4

- **Метод**: ToggleMinus()
- **Входные данные**: Объект класса TEditor вида 12,36-(i*12,35).
- **Ожидаемые данные**: 12,36+(i*12,35).

13. AddNumber_1

- **Метод**: AddNumber()
- **Входные данные**: Объект класса TEditor вида 0,36+(i*1,4).
- **Ожидаемые данные**: 4,36+(i*1,4).

14. AddNumber_2

- **Метод**: AddNumber()
- **Входные данные**: Объект класса TEditor вида -25,6-(i*44,44).
- **Ожидаемые данные**: -25,60-(i*44,44).

15. AddNumber_3

- **Метод**: AddNumber()
- **Входные данные**: Объект класса TEditor вида -25,6-(i*44,44).
- **Ожидаемые данные**: -25,6-(i*442,44).

16. AddNumber_4

- **Метод**: AddNumber()
- **Входные данные**: Объект класса TEditor вида -25,6-(i*44,44).
- **Ожидаемые данные**: -25,6-(i*44,445).

17. DelNumber_1

- **Метод**: DelNumber()
- **Входные данные**: Объект класса TEditor вида 5,4+(i*44,44).

- **Ожидаемые данные**: 0,4+(i*44,44).

18. DelNumber_2

- **Метод**: DelNumber()
- **Входные данные**: Объект класса TEditor вида 55,55-(i*3,3).
- **Ожидаемые данные**: 55,5-(i*3,3).

19. DelNumber_3

- **Метод**: DelNumber()
- **Входные данные**: Объект класса TEditor вида 24,03-(i*3,3).
- **Ожидаемые данные**: 24,03-(i*0,3).

20. DelNumber_4

- **Метод**: DelNumber()
- **Входные данные**: Объект класса TEditor вида 90,1+(i*5,97).
- **Ожидаемые данные**: 90,1+(i*5,9).

21. Clear_1

- **Метод**: Clear()
- **Входные данные**: Объект класса TEditor вида 55,55-(i*3,3).
- **Ожидаемые данные**: 0,-(i*0,).

22. AddZero_1

- **Метод**: AddZero()
- **Входные данные**: Объект класса TEditor вида 92,36+(i*1,4).
- **Ожидаемые данные**: 920,36+(i*1,4).

23. AddZero_2

- **Метод**: AddZero()

- **Входные данные**: Объект класса TEditor вида 0,02+(i*0,01).
- **Ожидаемые данные**: 0,020+(i*0,01).

24. AddZero_3

- **Метод**: AddZero()
- **Входные данные**: Объект класса TEditor вида 8,201+(i*6,9).
- **Ожидаемые данные**: 8,201+(i*60,9).

25. AddZero_4

- **Метод**: AddZero()
- **Входные данные**: Объект класса TEditor вида 3,0+(i*32,901).
- **Ожидаемые данные**: 3,0+(i*32,9010).

Вывод

В результате работы над лабораторной работой были сформированы практические навыки разработки функций классов на языке C#, разработка модульных тестов для тестирования функций классов и выполнения модульного тестирования на языке C# с помощью средств автоматизации Visual Studio.

Листинг программы:

Program.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace lab6
{
    class Program
    {
        static void Main(string[] args)
        {
            TEditor num1 = new TEditor();
            TEditor num2 = new TEditor();
            TEditor num3 = new TEditor();

            string number1 = "-1,36+(i*3,08)";
            string number2 = "43,81-(i*2,5)";

            num1.WriteNumber(number1);
            num2.WriteNumber(number2);
            Console.WriteLine($"Исходное число num1: {num1.ReadNumber()}");
            num1.ToggleMinus();
            Console.WriteLine($"Удаление минуса в действительной части num1:
{num1.ReadNumber()}");
            num1.DelNumber();
            Console.WriteLine($"Удаление символа в действительной части num1:
{num1.ReadNumber()}");
            num1.AddNumber(8);
            Console.WriteLine($"Добавлние цифры в действительную часть num1:
{num1.ReadNumber()}");
            num1.AddNumber(3);
            Console.WriteLine($"Добавлние ещё одной цфиры в действительную
часть num1: {num1.ReadNumber()}");
            num1.ToggleNumberMode();
            num1.DelNumber();
            Console.WriteLine($"Удаление цифры из действительной дробной части
num1: {num1.ReadNumber()}");
            num1.ToggleMode();
            num1.ToggleMinus();
            Console.WriteLine($"Измнение знака мнимой части на противоположный
num1: {num1.ReadNumber()}");
            num1.ToggleMinus();
            Console.WriteLine($"Изменение на исходный знак мнимой части num1:
{num1.ReadNumber()}");
            num1.ToggleNumberMode();
            num1.DelNumber();
            Console.WriteLine($"Удаление из мнимой части цифры в целой части
num1: {num1.ReadNumber()}");
            num1.ToggleNumberMode();
            num1.AddNumber(4);
            Console.WriteLine($"Добавление цифры в мнимую дробную часть num1:
{num1.ReadNumber()}");
            num1.AddNumber(5);
```

```
            Console.WriteLine($"Добавление ещё одной цифры в мнимую дробную
часть num1: {num1.ReadNumber()}");
            Console.WriteLine();

            Console.WriteLine($"Исходное число num2: {num2.ReadNumber()}");
            Console.WriteLine($"num2 нулевое: " + (num2.IsZero() ? "да" :
"нет"));
            num2.ToggleMode();
            num2.ToggleMinus();
            Console.WriteLine($"Измнение знака мнимой части num2:
{num2.ReadNumber()}");
            num2.Clear();
            Console.WriteLine($"Очищенное значение num2: {num2.ReadNumber()}");
            Console.WriteLine($"num2 нулевое: " + (num2.IsZero() ? "да" :
"нет"));


            num3.ShowEditInfo();
            Console.WriteLine("Нажмите на клавишу Esc для завершения.");
            while (true)
            {
                ConsoleKeyInfo key = Console.ReadKey(true);
                if(key.Key == ConsoleKey.Escape)
                {
                    break;
                }
                int command = -1;
                switch (key.Key)
                {
                    case ConsoleKey.D0:
                        command = 0;
                        break;
                    case ConsoleKey.D1:
                        command = 1;
                        break;
                    case ConsoleKey.D2:
                        command = 2;
                        break;
                    case ConsoleKey.D3:
                        command = 3;
                        break;
                    case ConsoleKey.D4:
                        command = 4;
                        break;
                    case ConsoleKey.D5:
                        command = 5;
                        break;
                    case ConsoleKey.D6:
                        command = 6;
                        break;
                    case ConsoleKey.D7:
                        command = 7;
                        break;
                    case ConsoleKey.D8:
                        command = 8;
                        break;
                }

                if (command != -1)
```

```
                    {
                        num3.Edit(command);
                    }
                }
            }
        }
    }
```

TEditor.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace lab6
{
    public enum PartToEdit
    {
        Real, Imag
    };
    public enum NumberPartToEdit
    {
        Left, Right
    };

    public class TEditor
    {
        string pNum;
        PartToEdit mode;
        NumberPartToEdit numberMode;
        string zero = "0,+(i*0,)";
        string separatorParts = "(i*";
        string separatorNumber = ",";

        public TEditor()
        {
            pNum = zero;
            mode = PartToEdit.Real;
            numberMode = NumberPartToEdit.Left;
        }

        public bool IsZero()
        {
            string tmp = pNum;
            if (tmp[0] == '-')
                tmp = tmp.Substring(1);
            tmp = tmp.Replace('-', '+');
            if (tmp == zero)
                return true;
            else
                return false;
        }

        public string ToggleMinus()
        {
            if (mode == PartToEdit.Real)
            {
                if (pNum[0] == '-')
                    pNum = pNum.Substring(1);
                else
                    pNum = '-' + pNum;
            }
            else
            {
                int separatorIndex = pNum.IndexOf(separatorParts);
```

```
                if (pNum[separatorIndex-1] == '-'){
                    pNum = pNum.Substring(0, pNum.IndexOf(separatorParts) - 1)
+ "+" +
                       pNum.Substring(pNum.IndexOf(separatorParts));
                }
                else if(pNum[separatorIndex-1] == '+')
                {
                    pNum = pNum.Substring(0, pNum.IndexOf(separatorParts) - 1)
+ "-" +
                       pNum.Substring(pNum.IndexOf(separatorParts));
                }
            }
            return pNum;
        }

        public PartToEdit ToggleMode()
        {
            if (mode == PartToEdit.Real)
                mode = PartToEdit.Imag;
            else
                mode = PartToEdit.Real;
            return mode;
        }

        public NumberPartToEdit ToggleNumberMode()
        {
            if (numberMode == NumberPartToEdit.Left)
                numberMode = NumberPartToEdit.Right;
            else
                numberMode = NumberPartToEdit.Left;
            return numberMode;
        }

        public string AddNumber(int a)
        {
            if (a < 0 || a > 9)
                return pNum;
            int ind = pNum.IndexOf(separatorParts);
            if (mode == PartToEdit.Real)
            {
                if (numberMode == NumberPartToEdit.Left)
                {
                    if (pNum[0] == '0')
                        pNum = a + pNum.Substring(1);
                    else if (pNum[0] == '-' && pNum[1] == '0')
                        pNum = '-' + a + pNum.Substring(2);
                    else
                    {
                        int frstNumbSep = pNum.IndexOf(separatorNumber);
                        pNum = pNum.Insert(frstNumbSep, a.ToString());
                    }
                }
                else
                    pNum = pNum.Insert(ind - 1, a.ToString());
            }
            else
            {
                if (numberMode == NumberPartToEdit.Left)
                {
```

```
                    ind += 2;
                    if (pNum[ind] == '0')
                        pNum = pNum.Substring(0, ind - 1) + a +
pNum.Substring(ind + 1);
                    else
                    {
                        int lastNumbSep = pNum.LastIndexOf(',');
                        pNum = pNum.Insert(lastNumbSep, a.ToString());
                    }
                }
                else
                {
                    int separatorIndex = pNum.LastIndexOf(")");
                    pNum = pNum.Insert(separatorIndex, a.ToString());
                }

            }
            return pNum;
        }

        public string AddZero()
        {
            return AddNumber(0);
        }

        public string DelNumber()
        {
            int ind = pNum.IndexOf(separatorParts);
            if (mode == PartToEdit.Real)
            {
                if (numberMode == NumberPartToEdit.Left)
                {
                    if (pNum[0] == '0')
                        return pNum;
                    else if (pNum[0] == '-' && pNum[1] == '0')
                        return pNum;
                    else
                    {
                        int frstNumbSep = pNum.IndexOf(separatorNumber);
                        pNum = pNum.Remove(frstNumbSep - 1, 1);
                        if (pNum[0] == ',')
                            pNum = '0' + pNum;
                    }
                }
                else
                {
                    int r = 0;
                    if (!int.TryParse(pNum[ind - 2].ToString(), out r))
                        return pNum;
                    pNum = pNum.Remove(ind - 2, 1);
                }
            }
            else
            {
                if (numberMode == NumberPartToEdit.Left)
                {
                    ind += 3;
                    if (pNum[ind] == '0')
                        return "0";
```

```
                    else
                    {
                        int lastNumbSep = pNum.LastIndexOf(',');
                        if (pNum[lastNumbSep - 2] == '*')
                            pNum = pNum.Substring(0, lastNumbSep - 1) + '0' +
pNum.Substring(lastNumbSep);
                        else
                            pNum = pNum.Remove(lastNumbSep - 1, 1);
                    }
                }
                else
                {
                    if (pNum[pNum.Length - 2] == ',')
                        return pNum;
                    else
                        pNum = pNum.Remove(pNum.Length - 2, 1);
                }
            }
            return pNum;
        }

        public string Clear()
        {
            pNum = zero;
            mode = PartToEdit.Real;
            numberMode = NumberPartToEdit.Left;
            return pNum;
        }

        public void ShowEditInfo()
        {
            Console.WriteLine("\nВведите число:\n" +
                "0 - Изменить знак на противоположный\n" +
                "1 - Добавить цифру\n" +
                "2 - Добавить ноль\n" +
                "3 - Удалить цифру\n" +
                "4 - Очистить комплексное число\n" +
                "5 - Записать комплексное число\n" +
                "6 - Изменить режим работы между действительной и мнимой
частью\n" +
                "7 - Изменить режим работы между целой и дробной частью\n" +
                "8 - Показать комплексное число\n");
        }
        public string Edit(int command)
        {
            switch (command)
            {
                case 0:
                    ToggleMinus();
                    Console.WriteLine($"Комплексное число с измененным знаком:
{this.ReadNumber()}");
                    break;
                case 1:
                    {
                        Console.Write("Число для добавления: ");
                        int num;
                        string input = Console.ReadLine();
                        num = int.Parse(input);
                        AddNumber(num);
```

```
                        Console.WriteLine($"Комплексное число с добавленным
цифрой: {this.ReadNumber()}");
                        break;
                    }
                case 2:
                    AddZero();
                    Console.WriteLine($"Комплексное число с добавленным нулём:
{this.ReadNumber()}");
                    break;
                case 3:
                    DelNumber();
                    Console.WriteLine($"Комплексное число с удаленной цифрой:
{this.ReadNumber()}");
                    break;
                case 4:
                    Clear();
                    Console.WriteLine($"Очищенное комплексное число:
{this.ReadNumber()}");
                    break;
                case 5:
                    {
                        Console.WriteLine("Введите комплексное число: ");
                        string inp;
                        inp = Console.ReadLine();
                        WriteNumber(inp);
                        Console.WriteLine($"Ваше комплексное число:
{this.ReadNumber()}");
                        break;
                    }
                case 6:
                    Console.WriteLine("Режим работы между действительной и
мнимой частью изменен.");
                    this.ToggleMode();
                    break;
                case 7:
                    Console.WriteLine("Режим работы между целой и дробной
частью изменен.");
                    this.ToggleNumberMode();
                    break;
                case 8:
                    Console.WriteLine($"Показываю комплексное число:
{this.ReadNumber()}");
                    break;

                default:
                    break;
            }
            return pNum;
        }

        public string WriteNumber(string otherNumber)
        {
            pNum = otherNumber;
            return pNum;
        }

        public string ReadNumber()
        {
            return pNum;
```

```
                }
            }
        }
```

UnitTest1.cs:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using lab6;

namespace lab6Tests
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Init_And_Output_1()
        {
            TEditor testClass = new TEditor();
            string output = "10,3+(i*0,8)";
            testClass.WriteNumber(output);
            Assert.AreEqual(output, testClass.ReadNumber());
        }
        [TestMethod]
        public void Init_And_Output_2()
        {
            TEditor testClass = new TEditor();
            string output = "-12,6-(i*66,2)";
            testClass.WriteNumber(output);
            Assert.AreEqual(output, testClass.ReadNumber());
        }
        [TestMethod]
        public void Init_And_Output_3()
        {
            TEditor testClass = new TEditor();
            string output = "0,3+(i*0,0)";
            testClass.WriteNumber(output);
            Assert.AreEqual(output, testClass.ReadNumber());
        }
        [TestMethod]
        public void Init_And_Output_4()
        {
            TEditor testClass = new TEditor();
            string output = "0,+(i*0,)";
            Assert.AreEqual(output, testClass.ReadNumber());
        }
        [TestMethod]
        public void IsZero_1()
        {
            TEditor testClass = new TEditor();
            testClass.WriteNumber("12,36+(i*12,35)");
            Assert.IsFalse(testClass.IsZero());
        }
        [TestMethod]
        public void IsZero_2()
        {
            TEditor testClass = new TEditor();
            testClass.WriteNumber("0,+(i*0,)");
            Assert.IsTrue(testClass.IsZero());
        }
        [TestMethod]
        public void IsZero_3()
        {
```

```
    TEditor testClass = new TEditor();
    testClass.WriteNumber("0,+(i*12,54)");
    Assert.IsFalse(testClass.IsZero());
}
[TestMethod]
public void IsZero_4()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("0,43+(i*0,)");
    Assert.IsFalse(testClass.IsZero());
}
[TestMethod]
public void ToggleMinus_1()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("12,36+(i*12,35)");
    testClass.ToggleMinus();
    string output = "-12,36+(i*12,35)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void ToggleMinus_2()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("-12,36+(i*12,35)");
    testClass.ToggleMinus();
    string output = "12,36+(i*12,35)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void ToggleMinus_3()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("12,36+(i*12,35)");
    testClass.ToggleMode();
    testClass.ToggleMinus();
    string output = "12,36-(i*12,35)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void ToggleMinus_4()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("12,36-(i*12,35)");
    testClass.ToggleMode();
    testClass.ToggleMinus();
    string output = "12,36+(i*12,35)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void AddNumber_1()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("0,36+(i*1,4)");
    testClass.AddNumber(4);
    string output = "4,36+(i*1,4)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
```

```
public void AddNumber_2()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("-25,6-(i*44,44)");
    testClass.ToggleNumberMode();
    testClass.AddNumber(0);
    string output = "-25,60-(i*44,44)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void AddNumber_3()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("-25,6-(i*44,44)");
    testClass.ToggleMode();
    testClass.AddNumber(2);
    string output = "-25,6-(i*442,44)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void AddNumber_4()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("-25,6-(i*44,44)");
    testClass.ToggleMode();
    testClass.ToggleNumberMode();
    testClass.AddNumber(5);
    string output = "-25,6-(i*44,445)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void DelNumber_1()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("5,4+(i*44,44)");
    testClass.DelNumber();
    string output = "0,4+(i*44,44)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void DelNumber_2()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("55,55-(i*3,3)");
    testClass.ToggleNumberMode();
    testClass.DelNumber();
    string output = "55,5-(i*3,3)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void DelNumber_3()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("24,03-(i*3,3)");
    testClass.ToggleMode();
    testClass.DelNumber();
    string output = "24,03-(i*0,3)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
```

```
[TestMethod]
public void DelNumber_4()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("90,1+(i*5,97)");
    testClass.ToggleMode();
    testClass.ToggleNumberMode();
    testClass.DelNumber();
    string output = "90,1+(i*5,9)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void Clear_1()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("55,55-(i*3,3)");
    testClass.Clear();
    string output = "0,+(i*0,)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void AddZero_1()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("92,36+(i*1,4)");
    testClass.AddZero();
    string output = "920,36+(i*1,4)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void AddZero_2()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("0,02+(i*0,01)");
    testClass.ToggleNumberMode();
    testClass.AddZero();
    string output = "0,020+(i*0,01)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void AddZero_3()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("8,201+(i*6,9)");
    testClass.ToggleMode();
    testClass.AddZero();
    string output = "8,201+(i*60,9)";
    Assert.AreEqual(output, testClass.ReadNumber());
}
[TestMethod]
public void AddZero_4()
{
    TEditor testClass = new TEditor();
    testClass.WriteNumber("3,0+(i*32,901)");
    testClass.ToggleMode();
    testClass.ToggleNumberMode();
    testClass.AddZero();
    string output = "3,0+(i*32,9010)";
    Assert.AreEqual(output, testClass.ReadNumber());
```

```
                }
            }
        }
```