

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

Лабораторная работа №10
«Редактор комплексных чисел»

Выполнил:
Студент группы ИП-013
Копытина Т.А.
Работу проверил:
ассистент кафедры ПМиК
Агалаков А.А.

Новосибирск 2023 г.

Содержание

1. Задание	3
2. Исходный код программы.....	5
2.1. Код программы	5
2.2. Код тестов.....	9
3. Результаты модульных тестов	11
4. Вывод	12

1. 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.
3. Если необходимо, предусмотрите возбуждение исключительных ситуаций. На Унифицированном языке моделирования UML (Unified Modeling Language) наш класс можно обозначить следующим образом:

РедакторКомплексныхЧисел

- строка: String
- комплексноеЧислоЕстьНоль: Boolean
- добавитьЗнак: String
- добавитьЦифру(a: Integer): String
- добавитьНоль: String
- забойСимвола: String
- очистить: String
- конструктор
- читатьСтрокаВформатеСтроки: String (метод свойства)
- писатьСтрокаВформатеСтроки(a: String) (метод свойства)
- редактировать(a: Integer): String

Обязанность:

ввод, хранение и редактирование строкового представления
комплексных чисел

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.

2. Исходный код программы

2.1. Код программы

TEditor.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace lab10
{
    public enum PartToEdit
    {
        Real, Imag
    };
    public enum NumberPartToEdit
    {
        Left, Right
    };

    public abstract 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
            {
                pNum = pNum.Substring(0,
pNum.IndexOf(separatorParts)) + "-" +
pNum.Substring(pNum.IndexOf(separatorPart
s));
            }
            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.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.Insert(lastNumbSep, a.ToString());
                }
            }
            else
                pNum += '0' + a;
        }
        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 += 2;
            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.Remove(lastNumbSep - 1, 1);
            }
        }
        else
        {
            if (pNum[pNum.Length
- 1] == ',')
                return pNum;
            else
                pNum.Remove(pNum.Length - 1);
        }
        return pNum;
    }

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

    public string Edit(int command)
    {
        switch (command)
        {
            case 0:
                ToggleMinus();
                break;
            case 1:
                {
                    Console.WriteLine("Enter number to add: ");
                    int num;
                    num =
Console.Read();

                    AddNumber(num);
                    break;
                }
            case 2:
                AddZero();
                break;
        }
    }

```

```

        case 3:
            DelNumber();
            break;
        case 4:
            Clear();
            break;
        case 5:
        {
Console.WriteLine("Enter string to write:
");
            string inp;
            inp =
Console.ReadLine();
            WriteNumber(inp);
            break;
        }
        default:

```

```

            break;
        }
        return pNum;
    }

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

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

```

Editor.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace lab9
{
    public class Editor : TEditor
    {
        public Editor() : base()
        {
        }
    }
}
```


2.2. Код тестов

TestComplexTEdit.cs

```
using
Microsoft.VisualStudio.TestTools.UnitTesting;
using lab10;

namespace TestComplexTEdit
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Init_And_Output_1()
        {
            Editor testClass = new
Editor();
            string output = "10,3+i*0,8";

testClass.WriteNumber(output);
            Assert.AreEqual(output,
testClass.ReadNumber());
        }
        [TestMethod]
        public void Init_And_Output_2()
        {
            Editor testClass = new
Editor();
            string output = "-12,6-
i*66,2";

testClass.WriteNumber(output);
            Assert.AreEqual(output,
testClass.ReadNumber());
        }
        [TestMethod]
        public void Init_And_Output_3()
        {
            Editor testClass = new
Editor();
            string output = "0,3+i*0,0";

testClass.WriteNumber(output);
            Assert.AreEqual(output,
testClass.ReadNumber());
        }
        [TestMethod]
        public void IsZero_1()
        {
            Editor testClass = new
Editor();

testClass.WriteNumber("12,36+i*12,35");

Assert.IsFalse(testClass.IsZero());
        }
        [TestMethod]
        public void IsZero_2()
        {
            Editor testClass = new

testClass.WriteNumber("0,+i*0,");

Assert.IsTrue(testClass.IsZero());
        }
        [TestMethod]
        public void ToggleMinus_1()
        {
            Editor testClass = new
Editor();

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()
        {
            Editor testClass = new
Editor();
            testClass.WriteNumber("-
12,36+i*12,35");
            testClass.ToggleMinus();
            string output =
"12,36+i*12,35";
            Assert.AreEqual(output,
testClass.ReadNumber());
        }
        [TestMethod]
        public void AddNumber_1()
        {
            Editor testClass = new
Editor();

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()
        {
            Editor testClass = new
Editor();
            testClass.WriteNumber("-25,6-
i*44,44");
            testClass.AddNumber(0);
            string output = "-250,6-
i*44,44";
            Assert.AreEqual(output,
testClass.ReadNumber());
        }
        [TestMethod]
        public void DelNumber_1()
        {

```

```

        Editor testClass = new
Editor();
testClass.WriteNumber("0,4+i*44,44");
    testClass.DelNumber();
    string output =
"0,4+i*44,44";
    Assert.AreEqual(output,
testClass.ReadNumber());
    }
[TestMethod]
public void DelNumber_2()
{
    Editor testClass = new
Editor();
testClass.WriteNumber("55,55-
i*3,3");
testClass.ToggleMode();
testClass.DelNumber();

```

```

        string output = "55,55-
i*0,3";
    Assert.AreEqual(output,
testClass.ReadNumber());
    }
[TestMethod]
public void Clear_1()
{
    Editor testClass = new
Editor();
testClass.WriteNumber("55,55-
i*3,3");
testClass.Clear();
    string output = "0,+i*0,";
    Assert.AreEqual(output,
testClass.ReadNumber());
    }
}
}

```

3. Результаты модульных тестов

Обозреватель тестов			
<div>▶ ▶ ▶ ↺ ⌂ 12 12 0 🔍 [⌵] [⌵] [⌵] ⚙</div>			
Запуск тестов завершен: тестов запущено в 318 мс: 12 (пройдено: 12, не пройдено: 0, пропущено: 0).			
Тестирование	Длительн...	Признаки	Сообщение об ошибке
▶ ✓ TestComplexTEdit (12)	79 мс		
▶ ✓ TestComplexTEdit (12)	79 мс		
▶ ✓ UnitTest1 (12)	79 мс		
✓ ToggleMinus_2	< 1 мс		
✓ ToggleMinus_1	< 1 мс		
✓ IsZero_2	< 1 мс		
✓ IsZero_1	< 1 мс		
✓ Init_And_Output_3	< 1 мс		
✓ Init_And_Output_2	< 1 мс		
✓ Init_And_Output_1	< 1 мс		
✓ DelNumber_2	< 1 мс		
✓ DelNumber_1	< 1 мс		
✓ Clear_1	< 1 мс		
✓ AddNumber_2	< 1 мс		
✓ AddNumber_1	79 мс		

4. Вывод

По итогам данной лабораторной работе были сформированы практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов C# и их модульного тестирования.