

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 №9
«Практическая работа. Редактор простых дробей»

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.....	8
3. Результаты модульных тестов.....	10
4. Вывод	11

1. Задание

1. Разработать и реализовать класс TEditor «Ввод и редактирование простых дробей», используя класс C++.

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

РедакторПростыхДробей

- строка: String
- дробьЕстьНоль: 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 простых дробей.

2. Класс должен отвечать за посимвольный ввод, хранение и редактирование строкового представления простых дробей. Значение нуля - '0|1'. Класс должен обеспечивать:

- добавление цифры;
- добавление и изменение знака;
- добавление разделителя целой и дробной частей;
- забой символа, стоящего справа (BackSpace);

- установку нулевого значения числа (Clear);
- чтение строкового представления простой дроби;
- запись строкового представления простой дроби.

3. Протестировать каждый метод класса.

2. Исходный код программы

2.1. Код программы

TEditor.cs

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

namespace lab9
{
    public abstract class TEditor
    {
        const string separator = "/";
        const string zeroStr = "0/1";
        string frac;

        public TEditor()
        {
            frac = zeroStr;
        }

        public bool IsZero()
        {
            if (frac[0] == zeroStr[0] ||
                (frac[0] == '-' && frac[1] ==
                 zeroStr[0]))
                return true;
            else return false;
        }

        public string ToggleMinus()
        {
            if (frac[0] == '-')
                frac = frac.Substring(1);
            else
                frac = '-' + frac;
            return frac;
        }

        public string AddNumber(int a)
        {
            if (a < 0 || a > 9)
                return frac;
            if (frac[0] != '-' && frac[0]
                != '0')
                frac = a +
                frac.Substring(2);
            else if (frac[0] == '-' &&
                frac[1] != '0')
                frac = "-" + a +
                frac.Substring(1);
            else if (frac[0] == '-' &&
                frac[1] == '0')
                frac = "-" + a +
                frac.Substring(2);
            else if (frac[0] == '0')
                frac = a +
                frac.Substring(1);
            return frac;
        }

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

        public string DelSymbol()
        {
            if (frac[frac.Length - 2] !=
                separator[0])
            {
                frac = frac.Substring(0,
                    frac.Length - 1);
            }

            return frac;
        }

        public string Clear()
        {
            frac = zeroStr;
            return frac;
        }

        string Edit(int command)
        {
            switch (command)
            {
                case 0:
                    this.ToggleMinus();
                    break;
                case 1:
                    {
                        Console.Write("Enter number to add:");
                        int num;
                        num =
                            Console.Read();
                        AddNumber(num);
                        break;
                    }
                case 2:
                    this.AddZero();
                    break;
                case 3:
                    this.DelSymbol();
                    break;
                case 4:
                    this.Clear();
                    break;
                case 5:
                    {
                        Console.Write("Enter string to write:");
                        string inp;
                        inp =
                            Console.ReadLine();
                        this.WriteString(inp);
                        break;
                    }
            }
        }
    }
}
```

```

        }
        default:
            return frac;
    }
    return frac;
}

public string ReadString()
{
    return frac;
}

```

```

    }

    public string WriteString(string
otherStr)
    {
        frac = otherStr;
        return frac;
    }
}

```

Editor.cs

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

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

2.2. Код тестов

TestFracTEdit.cs

```
using
Microsoft.VisualStudio.TestTools.UnitTesting;
using lab9;

namespace TestFracTEdit
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Init_And_Output_1()
        {
            Editor testClass = new
Editor();
            string output = "3/4";

testClass.WriteString(output);
            Assert.AreEqual(output,
testClass.ReadString());
        }

        [TestMethod]
        public void Init_And_Output_2()
        {
            Editor testClass = new
Editor();
            string output = "-16/3";

testClass.WriteString(output);
            Assert.AreEqual(output,
testClass.ReadString());
        }

        [TestMethod]
        public void IsZero_1()
        {
            Editor testClass = new
Editor();

testClass.WriteString("14/3");

Assert.IsFalse(testClass.IsZero());
        }

        [TestMethod]
        public void IsZero_2()
        {
            Editor testClass = new
Editor();

testClass.WriteString("0/11");

Assert.IsTrue(testClass.IsZero());
        }

        [TestMethod]
        public void ToggleMinus_1()
        {
            Editor testClass = new
Editor();

testClass.WriteString("3/4");
testClass.ToggleMinus();
            string output = "-3/4";
            Assert.AreEqual(output,
testClass.ReadString());
        }

        [TestMethod]
        public void ToggleMinus_2()
        {
            Editor testClass = new
Editor();

testClass.WriteString("-
3/4");

testClass.ToggleMinus();
            string output = "3/4";
            Assert.AreEqual(output,
testClass.ReadString());
        }

        [TestMethod]
        public void AddNumber_1()
        {
            Editor testClass = new
Editor();

testClass.WriteString("-
3/4");

testClass.AddNumber(4);
            string output = "-43/4";
            Assert.AreEqual(output,
testClass.ReadString());
        }

        [TestMethod]
        public void AddNumber_2()
        {
            Editor testClass = new
Editor();

testClass.WriteString("0/1");
testClass.AddNumber(4);
            string output = "4/1";
            Assert.AreEqual(output,
testClass.ReadString());
        }

        [TestMethod]
        public void DelNumber_1()
        {
            Editor testClass = new
Editor();

testClass.WriteString("42/3");
testClass.DelSymbol();
            string output = "42/3";
            Assert.AreEqual(output,
testClass.ReadString());
        }
    }
}
```



```

[TestMethod]
public void DelNumber_2()
{
    Editor testClass = new
Editor();
testClass.WriteString("42/34");
    testClass.DelSymbol();
    string output = "42/3";
    Assert.AreEqual(output,
testClass.ReadString());
}

[TestMethod]

```

```

public void Clear()
{
    Editor testClass = new
Editor();
testClass.WriteString("42/34");
    testClass.Clear();
    string output = "0/1";
    Assert.AreEqual(output,
testClass.ReadString());
}
}
}

```

3. Результаты модульных тестов

Обозреватель тестов

Запуск тестов завершен: тестов запущено в 205 мс: 11 (пройдено: 11, не пройдено: 0, пропущено: 0).

Тестирование	Длительн...	Признаки	Сообщение об ошибке
TestProject1 (11)	8 мс		
TestFracTEdit (11)	8 мс		
UnitTest1 (11)	8 мс		
ToggleMinus_2	< 1 мс		
ToggleMinus_1	< 1 мс		
IsZero_2	< 1 мс		
IsZero_1	< 1 мс		
Init_And_Output_2	< 1 мс		
Init_And_Output_1	< 1 мс		
DelNumber_2	< 1 мс		
DelNumber_1	< 1 мс		
Clear	< 1 мс		
AddNumber_2	< 1 мс		
AddNumber_1	8 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