**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

по дисциплине: «Объектно-ориентированное проектирование и программирование»

на тему: **«**Обработка строковых данных. Регулярные выражения**»**

Выполнил: студент гр. ИТИ-21

Бахонько А.М.

Принял: преподаватель-стажёр

Башаримов Ю. С.

Гомель 2023

**Цель работы**: научится разрабатывать классы для решения задач и модульные тесты для тестирования созданного класса; уметь выполнять обработку строковых данных, обучится работе с регулярными выражениями.

**Задание:**

1. Разработать приложения, решающие поставленные задачи   
   (Вариант 2).
2. Разработать не менее 15 модульных тестов для тестирования.

**Ход работы**

Необходимо создать класс для работы со строками, чтобы составить в дальнейшем программу, которая будет вводить строку в переменную string. Внутри класса разработать метод, который создаст новую строку с индивидуальными символами введенной строки. Потом это строка будет выведена с количеством индивидуальных символов. Следующая задача заключается в том, что каждая строка текста содержит телефонный номер, время звонка и длительность звонка в секундах (если это не так, генерировать исключение), сформировать новую строку типа *StringBuilder* из строк, в которых звонки были на заданный номер после полуночи. Формат времени: чч-мм.

На рисунке 1 изображен случай при вводе пустой строки:

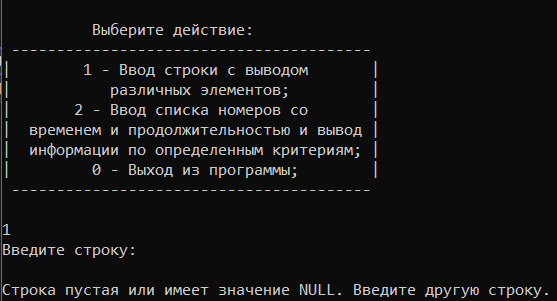


Рисунок 1 – Случай ввода пустой строки.

На рисунке 2 изображена работа поиска индивидуальных символов во введенной строке с выводом их:

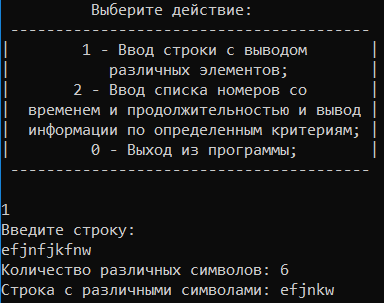


Рисунок 2 – Окно консоли при выполнении первой задачи.

Уже на рисунке 3 изображён результат, когда совершается попытка ввода значение, которое имеет не такой формат, который необходим:

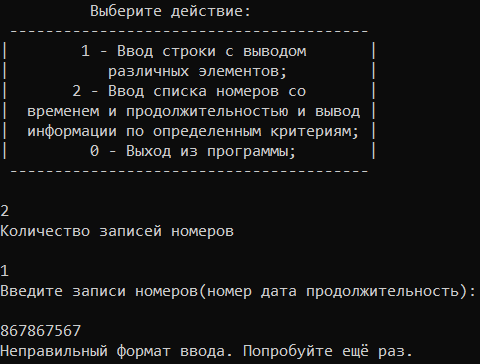


Рисунок 3 – Окно консоли при вводе данных в неправильном формате.

На рисунке 4 изображен результат работы при вводе данных в правильном формате:

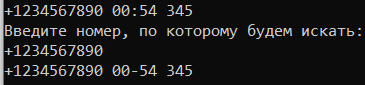


Рисунок 4 – Результаты работы действия 2.

В приложении А и Б представлен код программы и код библиотеки.

На рисунке 5 уже изображён результат тестов, разработанных для проверки операций, созданных внутри классов для работы со строками:

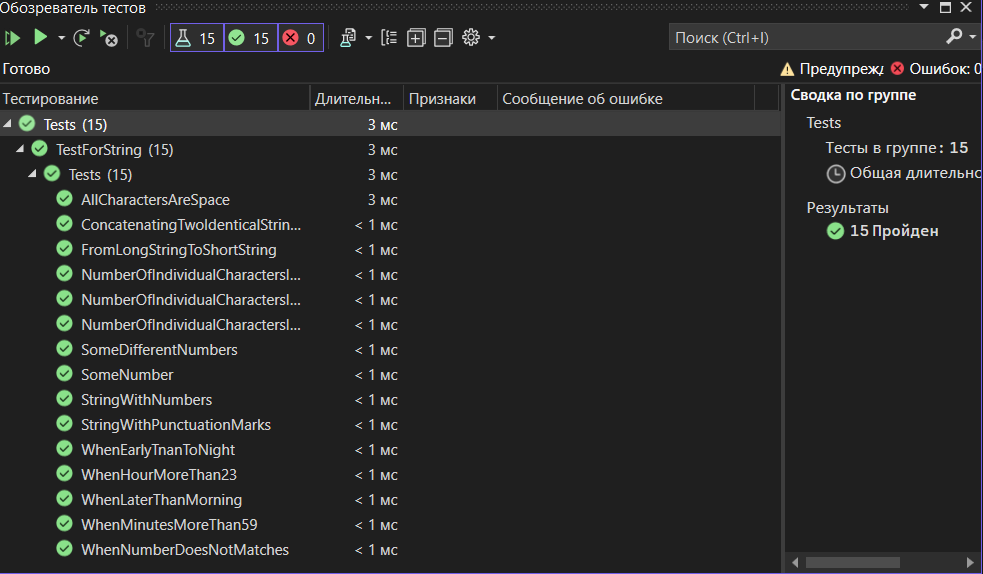


Рисунок 5 – 15 пройденных модульных тестов.

На рисунке 6 показан журнал *Git* для данной программы:

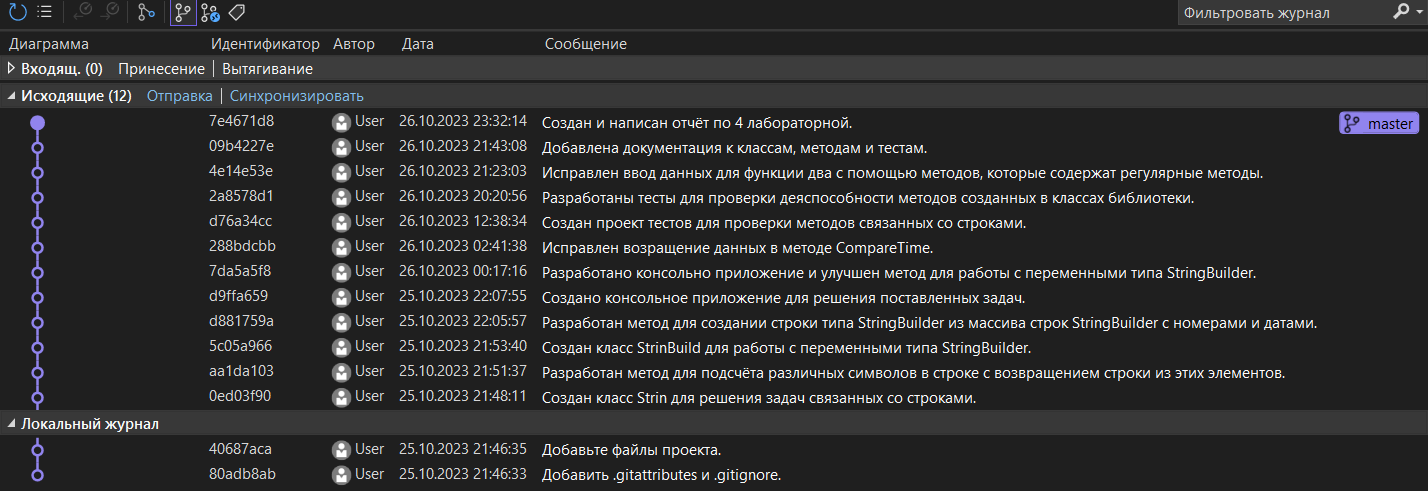


Рисунок 6 – Журнал *Git*.

Так как тесты все верны, то можно сделать вывод, что операции были разработаны правильно, а именно составления из индивидуальных символов строки новую строку и составление строки из строк содержащих номера телефон, время звонка и продолжительность в секундах, которые совпадают номерами с заданным и произведены после полуночи. Последний метод имеет тип возвращаемой переменной *StringBuilder*.

Регулярные выражения были использованы для создания формата ввода массива строк и строки, содержащей заданный номер.

Так как переменные типа *StringBuilder* нельзя сравнивать между собой в тестах мы их конвертируем в обычные строки.

В приложении В представлен код модульных тестов.

При выполнении данной работы были сделаны следующие вещи:

– созданы классы для обработки строк;

– создана и подключена к основной части библиотека;

– разработаны методы для выполнения поставленных задач;

– организовано консольное меню пользователя;

– использованы регулярные выражения для формирования исключений;

– выполнены модульные тесты;

**Вывод:** в результате выполнения лабораторной работы научились обрабатывать с помощью методов строки, также использовать тип *StringBuilder*, чтобы не создавать новые строки, изучили работу с регулярными выражениями, разработали модульные тесты для проверки методов.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программы**

**Program.cs:**

using ClassForStrings;

using System.Text;

using static System.Net.Mime.MediaTypeNames;

using System.Text.RegularExpressions;

namespace String

{

public class Program

{

static void Main(string[] args)

{

bool isRun = true; // Присваивание переменной IsRun значения true.

while (isRun) // Консольное меню.

{

Console.WriteLine("\n Выберите действие:");

Console.WriteLine(" ----------------------------------------");

Console.WriteLine("| 1 - Ввод строки с выводом |");

Console.WriteLine("| различных элементов; |");

Console.WriteLine("| 2 - Ввод списка номеров со |");

Console.WriteLine("| временем и продолжительностью и вывод |");

Console.WriteLine("| информации по определенным критериям; |");

Console.WriteLine("| 0 - Выход из программы; |");

Console.WriteLine(" ----------------------------------------\n");

switch (Convert.ToInt32(Console.ReadLine()))

{

case 1: // Вводим строку и выводим новую строку с индивидуальными символами.

Console.WriteLine("Введите строку: ");

string str = Console.ReadLine();

if (str != "")

{

string dif = Strin.DifferentCharacters(str);

Console.WriteLine("Количество различных символов: " + dif.Length);

Console.WriteLine("Строка с различными символами: " + dif);

}

else

{

Console.WriteLine("Строка пустая или имеет значение NULL. Введите другую строку.\n");

}

break;

case 2: //Ввод массива строчек с номерами, временем и продолжительностью и составления новой строки

Console.WriteLine("Количество записей номеров\n");

int n = Program.ReadInt(), i = 0;

Console.WriteLine("Введите записи номеров(номер дата продолжительность):\n");

string[] numbers = new string[n];

while(i < n)

{

string input = Console.ReadLine();

if (ValidateInput(input) )

{

string[] num = input.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

string hour = num[1][0] + "" + num[1][1];

string minutes = num[1][3] + "" + num[1][4];

if (Convert.ToInt32(hour) < 24 && Convert.ToInt32(minutes) < 60)

{

numbers[i] = input;

i++;

}

else

{

Console.WriteLine("Неправильный формат ввода. Попробуйте ещё раз.");

}

}

else

{

Console.WriteLine("Неправильный формат ввода. Попробуйте ещё раз.");

}

}

Console.WriteLine("Введите номер, по которому будем искать: ");

string inputnumber = Console.ReadLine(), number = "";

if (NumberInput(inputnumber))

{

number = inputnumber;

}

else

{

Console.WriteLine("Неправильный формат ввода. Попробуйте ещё раз.");

}

StringBuilder sb = new StringBuilder();

sb = StrinBuild.CompareTime(numbers, number);

Console.WriteLine(sb);

break;

case 0:

isRun = false; // Выход из программы.

break;

default:

Console.WriteLine("Некорректный выбор функции!");

break;

}

}

}

/// <summary>

/// Метод для проверки вводимого значения для целочисленного числа.

/// </summary>

/// <returns></returns>

public static int ReadInt()

{

string numeral = Console.ReadLine();

int value;

while (!Int32.TryParse(numeral, out value) && Convert.ToInt32(numeral) < 1)

{

Console.WriteLine("Вводные данные не подходят. Введите корректное значение: ");

numeral = Console.ReadLine();

}

return value;

}

/// <summary>

/// Метод проверки с помощью регулярного выражения значения для массива строк.

/// </summary>

/// <param name="input"></param>

/// <returns></returns>

static bool ValidateInput(string input)

{

string pattern = @"^\+\d+ \d{2}:\d{2} \d+$"; // Формат: +XXXXXXXXXXXXX 00:00 XXX

return Regex.IsMatch(input, pattern);

}

/// <summary>

/// Метод проверки с помощью регулярного выражения значения для строки номера.

/// </summary>

/// <param name="input"></param>

/// <returns></returns>

static bool NumberInput(string input)

{

string pattern = @"^\+\d+$"; //Формат: +XXXXXXXXXXXXX

return Regex.IsMatch(input, pattern);

}

}

}

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Текста библиотеки**

**Strin.cs:**

namespace ClassForStrings

{

/// <summary>

/// Класс для работы со строками.

/// </summary>

public class Strin

{

/// <summary>

/// Метод, которые позволяет найти в строке индивидуальные символы и составить из них новую строку.

/// </summary>

/// <param name="Str"></param>

/// <returns></returns>

public static string DifferentCharacters(string Str)

{

string dif = "";

int i = 1;

dif = dif + Str[0];

while (i < Str.Length)

{

int j = 0;

bool isContain = false;

while (j < dif.Length && !isContain)

{

if (dif[j] == Str[i])

{

isContain = true;

}

j++;

}

if (!isContain)

{

dif = dif + Str[i];

}

i++;

}

return dif;

}

}

}

**StrinBuild.cs:**

using System.Text;

namespace ClassForStrings

{

/// <summary>

/// Класс для работы с переменными типа StringBiulder.

/// </summary>

public class StrinBuild

{

/// <summary>

/// Метод для составления строки, в которой буду записаны все строки, что имеют заданный номер и совершенны с 00-00 и до 6-00

/// </summary>

/// <param name="numbers"></param>

/// <param name="number"></param>

/// <returns></returns>

public static StringBuilder CompareTime(string[] numbers, string number)

{

StringBuilder sb = new StringBuilder();

int i = 0;

while (i < numbers.Length)

{

string[] num = numbers[i].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

int j = 1;

string numb = "";

while (j < num[0].Length)

{

numb = numb + num[0][j];

j++;

}

j = 0;

string hour = num[1][0] + "" + num[1][1];

string minutes = num[1][3] + "" + num[1][4];

if (num[0].Length > 8 )

{

string newnumber, time = hour + "-" + minutes;

newnumber = num[0] + " " + time + " " + num[2] + "\n";

if (((Convert.ToInt32(hour) == 0 && Convert.ToInt32(minutes) > 0) || (Convert.ToInt32(hour) > 0 && Convert.ToInt32(hour) < 6) || (Convert.ToInt32(hour) == 6 && Convert.ToInt32(minutes) == 0)) && (num[0] == number))

{

sb.AppendJoin(" ", newnumber);

}

}

i++;

}

return sb;

}

}

}

**ПРИЛОЖЕНИЕ В**

(обязательное)

**Текст тестов**

**Tests.cs:**

using ClassForStrings;

using System.Text;

namespace TestForString

{

[TestClass]

public class Tests

{

/// <summary>

/// Тест, проверяющий вывод индивидуальных значений для строки запоненной только пробелами.

/// </summary>

[TestMethod]

public void AllCharactersAreSpace()

{

string string1 = " ";

string result = Strin.DifferentCharacters(string1);

string string2 = " ";

Assert.AreEqual(result, string2);

}

/// <summary>

/// Тест, проверяющий вывод индивидуальных символов для двух одинаковых строк.

/// </summary>

[TestMethod]

public void ConcatenatingTwoIdenticalStrings()

{

string string1 = "gfr";

string string2 = "gfr";

string result = Strin.DifferentCharacters(string1 + string2);

string string3 = "gfr";

Assert.AreEqual(result, string3);

}

/// <summary>

/// Тест, проверяющий вывод индивидуальных символов для длиной строки.

/// </summary>

[TestMethod]

public void FromLongStringToShortString()

{

string string1 = "fergrrrgrgrefgerfgerfger";

string result = Strin.DifferentCharacters(string1);

string string2 = "ferg";

Assert.AreEqual(result, string2);

}

/// <summary>

/// Тест, проверяющий вывод индивидуальных символов для строки со знаками пунктуации.

/// </summary>

[TestMethod]

public void StringWithPunctuationMarks()

{

string string1 = "():;, rtrt.rt";

string result = Strin.DifferentCharacters(string1);

string string2 = "():;, rt.";

Assert.AreEqual(result,string2);

}

/// <summary>

/// Тест, проверяющий вывод индивидуальных символов для строки с цифрами.

/// </summary>

[TestMethod]

public void StringWithNumbers()

{

string string1 = "1111222222333333334444444445555555555566666666677777";

string result = Strin.DifferentCharacters(string1);

string string2 = "1234567";

Assert.AreEqual(result,string2);

}

/// <summary>

/// Тест, проверяющий количество индвидуальных символов для длиных строк;.

/// </summary>

[TestMethod]

public void NumberOfIndividualCharactersInLongString()

{

string string1 = "qwertyuiopqwertyuiopqwertyuiopqwertyuiop";

string result = Strin.DifferentCharacters(string1);

int number = 10;

Assert.AreEqual(result.Length, number);

}

/// <summary>

/// Тест, проверяющий количество индивидуальных символов в строке с одним символом.

/// </summary>

[TestMethod]

public void NumberOfIndividualCharactersInStringWithOneCharacter()

{

string string1 = "w";

string result = Strin.DifferentCharacters(string1);

int number = 1;

Assert.AreEqual(result.Length,number);

}

/// <summary>

/// Тест, проверяющий количество индивидуальных символов для двух рахных строк.

/// </summary>

[TestMethod]

public void NumberOfIndividualCharactersInTwoDifferentStrings()

{

string string1 = "12345";

string string2 = "qwertsd";

string result = Strin.DifferentCharacters(string2 + string1);

int number = 12;

Assert.AreEqual(result.Length,number);

}

/// <summary>

/// Тест, проверяющий вывод строки, когда часов больше чем 23.

/// </summary>

[TestMethod]

public void WhenHourMoreThan23()

{

string[] numbers = { "+21832378237282728 24:12 232323" };

string number = "+21832378237282728";

StringBuilder result = StrinBuild.CompareTime(numbers,number);

StringBuilder sb = new StringBuilder();

Assert.AreEqual(result.ToString(),sb.ToString());

}

/// <summary>

/// Тест, проверяющий вывод строки, когда минут больше чем 59.

/// </summary>

[TestMethod]

public void WhenMinutesMoreThan59()

{

string[] numbers = { "+23232323232321 12:79 232" };

string number = "+23232323232321";

StringBuilder result = StrinBuild.CompareTime(numbers, number);

StringBuilder sb = new StringBuilder();

Assert.AreEqual(result.ToString(), sb.ToString());

}

/// <summary>

/// Тест, проверяющий вывод строки, когда время до полуночи.

/// </summary>

[TestMethod]

public void WhenEarlyTnanToNight()

{

string[] numbers = { "+321321321323 23:10 23212" };

string number = "+321321321323";

StringBuilder result = StrinBuild.CompareTime(numbers, number);

StringBuilder sb = new StringBuilder();

Assert.AreEqual(result.ToString(),sb.ToString());

}

/// <summary>

/// Тест, проверяющий вывод строки, когда время после утра.

/// </summary>

[TestMethod]

public void WhenLaterThanMorning()

{

string[] numbers = { "+32231232321232131 06:01 232132" };

string number = "+32231232321232131";

StringBuilder result = StrinBuild.CompareTime(numbers, number);

StringBuilder sb = new StringBuilder();

Assert.AreEqual(result.ToString(),sb.ToString());

}

/// <summary>

/// Тест, проверяющий вывод строки, когда номер не совпадает.

/// </summary>

[TestMethod]

public void WhenNumberDoesNotMatches()

{

string[] numbers = { "+213123 00:12 213232" };

string number = "+213122";

StringBuilder result = StrinBuild.CompareTime(numbers , number);

StringBuilder sb = new StringBuilder();

Assert.AreEqual(result.ToString(),sb.ToString());

}

/// <summary>

/// Тест, проверяющий вывод строки с объединением нескольких строк.

/// </summary>

[TestMethod]

public void SomeNumber()

{

string[] numbers = { "+677567576565 00:12 12", "+677567576565 01:12 26", "+677567576565 05:12 30" };

string number = "+677567576565";

StringBuilder result = StrinBuild.CompareTime(numbers , number);

StringBuilder sb = new StringBuilder();

sb.AppendJoin("", "+677567576565 00-12 12\n+677567576565 01-12 26\n+677567576565 05-12 30\n");

Assert.AreEqual(result.ToString(),sb.ToString());

}

/// <summary>

/// Тест, проверяющий вывод строки с объединением нескольких строк и исключением строк с другим номером.

/// </summary>

[TestMethod]

public void SomeDifferentNumbers()

{

string[] numbers = { "+67756757687 00:12 12", "+677567576565 01:12 26", "+677567576565 05:12 30" };

string number = "+677567576565";

StringBuilder result = StrinBuild.CompareTime(numbers, number);

StringBuilder sb = new StringBuilder();

sb.AppendJoin("", "+677567576565 01-12 26\n+677567576565 05-12 30\n");

Assert.AreEqual(result.ToString(), sb.ToString());

}

}

}