**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ЛАБОРАТОРНАЯ РАБОТА №5**

по дисциплине: «Объектно-ориентированное программирование»

на тему: «Парсинг текстовых данных»

Выполнил: студент гр. ИТП-21

Король В.Н.

Принял: преподаватель-стажер

Гуменников Е.Д.

Гомель 2022

**Цель работы:** изучить такой тип данных как строки на языке программирования *C*#. Изучить основные методы работы с ними, регулярные выражения для изменения и проверки строк и такой класс как StringBuilder для оптимизации работы со строками в *C*#. Так же изучить такое понятие как парсинг строк.

**Задание**

Необходимо решить задачу, согласно варианта (таблица 5)

2. При создании классов руководствоваться Code Convention

3. Весь код должен содержать элементы документирования

4. Разработать модульные тесты для верификации созданного класса

5. Класс должен быть размещен в библиотеке классов

6. Модульные тесты – в отдельном проекте

7. В отдельном проекте реализовать консольный интерфейс.

**Вариант 14**

В строках (класс StringBuilder) содержатся отчеты о произошедших ДТП. Из отчетов необходимо извлечь встречающиеся номера автомобилей и отсортировать их по странам регистрации.

**Ход Работы**

В ходе выполнения задания был создан класс *FindPlate* который выполняет обработку переданных строк при помощи регулярных выражений. После того как строки были обработаны они отбираются по условию задания и нужные нам добавляются в лист класса *StringBuilder.* Пример работы данного алгоритма указан на рисунке 1.

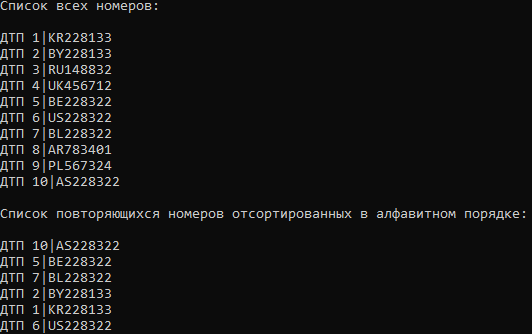


Рисунок 1 – Пример работы класса *FindPlate*

Для тестирования алгоритма были разработаны модульные тесты. Пример их работы казан на рисунке 2.

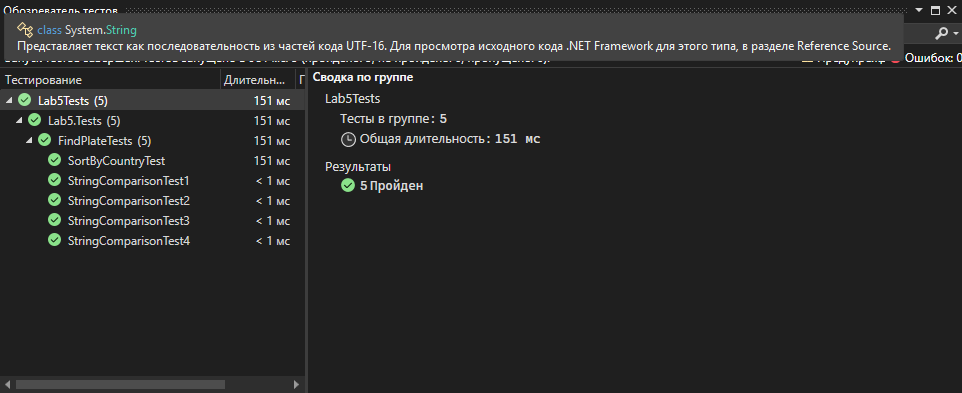


Рисунок 2 – Пример работы класса *FindPlate*

**Вывод:** в ходе выполнения лабораторной работы был изучен такой тип данных как строки на языке программирования *C*#. Основные методы работы с ними. Изучен такой класс как *StringBuilder* который оптимизирует работу со строками в *C*#. Также были выполнения задания в соответствии со своим вариантом.

**Листинг класса *FindPlate***

using System;

using System.Collections.Generic;

using System.Diagnostics.SymbolStore;

using System.Linq;

using System.Runtime.ExceptionServices;

using System.Runtime.InteropServices;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace Lab5

{

public class FindPlate

{

public List<StringBuilder> Plate = new List<StringBuilder>();

public FindPlate(string[] plate)

{

string pattern1 = @"[A-Z]{2}\d{6}";

for (int i = 0; i < plate.Length; i++)

{

if (Regex.IsMatch(plate[i].Split('|')[1], pattern1))

{

Plate.Add(new StringBuilder(plate[i]));

}

}

}

public List<StringBuilder> SortByCountry()

{

for(int i = 0; i < Plate.Count; i++)

{

int kol = 0;

for(int j = 0; j < Plate.Count; j++)

{

if (Regex.Replace(Plate[i].ToString().Split('|')[1], @"\D", "") == Regex.Replace(Plate[j].ToString().Split('|')[1], @"\D", ""))

{

kol++;

}

}

if(kol <= 1)

{

Plate.RemoveAt(i);

i = 0;

}

}

Console.WriteLine();

for(int i = 0; i < Plate.Count; i++)

{

for(int j = 0; j < Plate.Count; j++)

{

if (StringСomparison(Regex.Replace(Plate[i].ToString().Split('|')[1], @"\d", ""), Regex.Replace(Plate[j].ToString().Split('|')[1], @"\d", "")))

{

var buf = Plate[i];

Plate[i] = Plate[j];

Plate[j] = buf;

}

}

}

return Plate;

}

public bool StringСomparison(string str1, string str2)

{

int n = str1.Length > str2.Length ? (str1.Length) : (str2.Length);

for( int i = 0; i < n; i++)

{

if (str1[i] < str2[i])

{

return true;

}

else if (str1[i] > str2[i])

{

return false;

}

}

return false;

}

}

}

**Листинг тестов класса FindPlate**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Lab5;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab5.Tests

{

[TestClass()]

public class FindPlateTests

{

[TestMethod()]

public void SortByCountryTest()

{

FindPlate findPlate = new FindPlate(new string[] { "ДТП 1|FR222222", "ДТП 2|UK222222", "ДТП 3|BY101010" });

List<StringBuilder> act = findPlate.SortByCountry();

List<StringBuilder> expect = new List<StringBuilder> { new StringBuilder("ДТП 1|FR222222"), new StringBuilder("ДТП 2|UK222222") };

for(int i = 0; i < act.Count; i++)

{

Assert.AreEqual(act[i].ToString(), expect[i].ToString());

}

}

[TestMethod()]

public void StringСomparisonTest1()

{

string str1 = "abcde";

string str2 = "abced";

FindPlate act = new FindPlate(new string[] { "sdf|FR243456" });

Assert.IsTrue(act.StringСomparison(str1, str2));

}

[TestMethod()]

public void StringСomparisonTest2()

{

string str1 = "abcde";

string str2 = "abcde";

FindPlate act = new FindPlate(new string[] { "sdf|FR243456" });

Assert.IsFalse(act.StringСomparison(str2, str1));

}

[TestMethod()]

public void StringСomparisonTest3()

{

string str1 = "abcde";

string str2 = "abced";

FindPlate act = new FindPlate(new string[] { "sdf|FR243456" });

Assert.IsFalse(act.StringСomparison(str2, str1));

}

[TestMethod()]

public void StringСomparisonTest4()

{

string str1 = "aaa";

string str2 = "bbbbbbbbbbbbbbbb";

FindPlate act = new FindPlate(new string[] { "sdf|FR243456" });

Assert.IsTrue(act.StringСomparison(str1, str2));

}

}

}