**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 №5

по дисциплине: «Объектно-ориентированное проектирование и программирование»

на тему: **«**Парсинг текстовых данных**»**

Выполнил: студент гр. ИТИ-21

Бахонько А.М.

Принял: преподаватель

Башаримов Ю. С.

Гомель 2023

**Цель работы**: научится преобразовывать из неструктурированной записи данных в структурированную; разработать класс для реализации парсинга текстовых данных, проверить работу программы с помощью разработанных модульных текстов.

**Задание:**

1. Необходимо решить задачу, согласно варианта (Вариант 2).
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 размещён в библиотеке классов.
6. Модульные тесты – в отдельном проекте.
7. В отдельном проекте реализовать консольный интерфейс.

**Ход работы**

Необходимо создать класс для работы со строками типа *StringBuilder*, чтобы содержавшийся внутри текст, в котором встречаются *IP-*адреса компьютеров, преобразовать в массив строк, содержащих, соответственно, встречающиеся *IP*-адреса, рассортированные по своим классам.

На рисунке 1 изображен случай при вводе номера записей меньше 1:

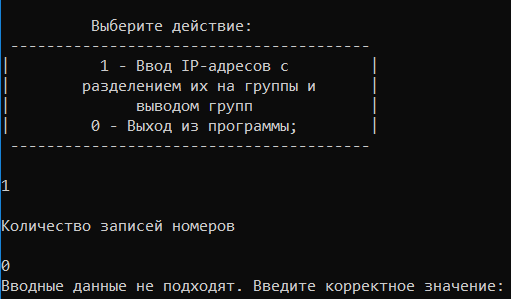


Рисунок 1 – Случай ввода количества записей меньше 1.

На рисунке 2 изображен случай при вводе в строку записей пустую строку:

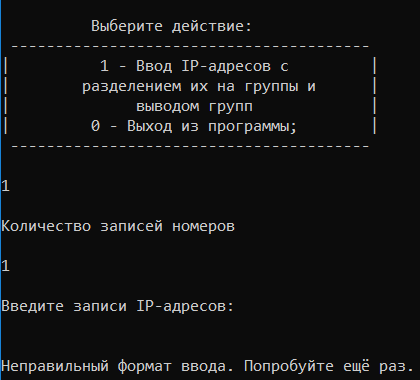


Рисунок 2 – Окно консоли при вводе пустой строки.

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

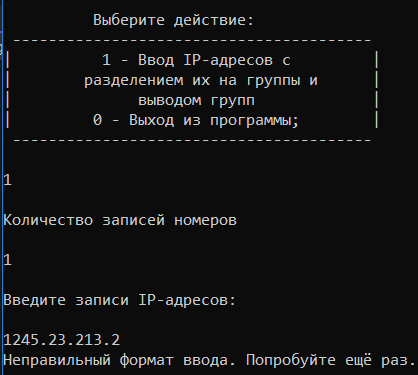


Рисунок 3 – Окно консоли при вводе данных в неправильном формате.

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

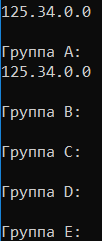


Рисунок 4 – Результаты работы метода для парсинга текста.

На рисунке 5 изображена работа программы при вводе нескольких записей в строку тип *StringBuilder*:

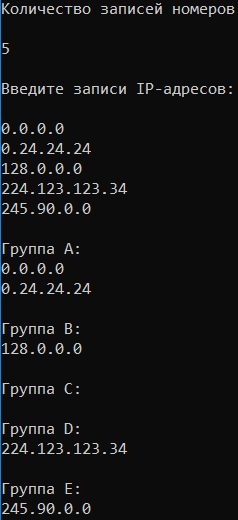


Рисунок 5 – Результат работы программы при вводе 5 записей.

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

На рисунке 6 уже изображён результат тестов, разработанных для проверки метода, созданного внутри класса для парсинга текстовых данных:

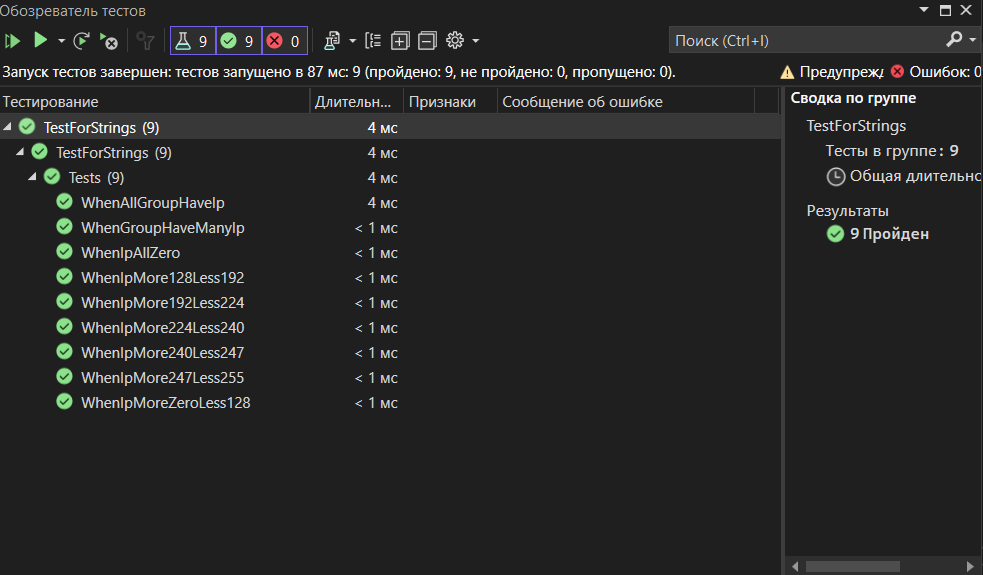


Рисунок 6 – 9 пройденных модульных тестов.

На рисунке 7 показан журнал *Git* для данной программы:

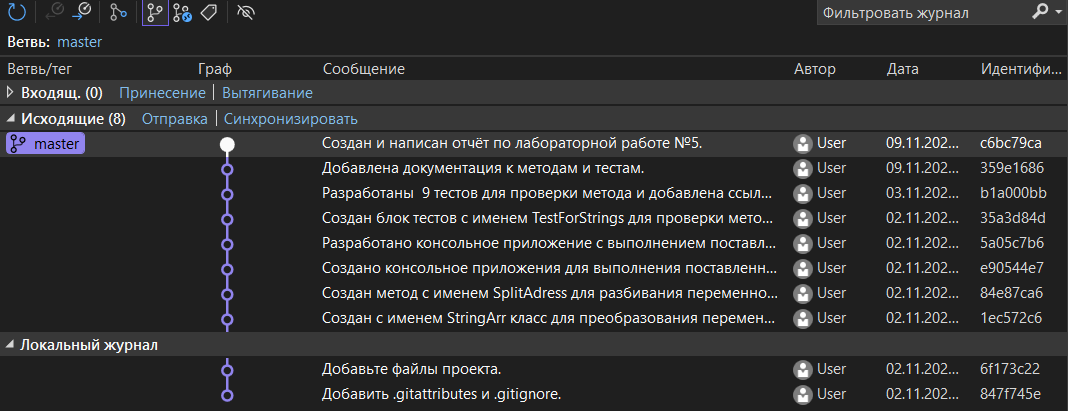


Рисунок 7 – Журнал *Git*.

Судя по тестам, можно сделать вывод, что метод был разработан верно, а именно составления из неструктурированной строки типа *StringBuilder* массив строк, разделённые по классам *IP*-адресов.

Регулярные выражения были использованы для создания формата ввода строки, содержащей значение *IP*-адреса.

В приложении В представлен код модульных тестов.

При выполнении данной работы были сделаны следующие вещи:

– создан класс для парсинга текстовых данных;

– 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 преобразовывать неструктурированную информацию в структурированную с помощью методов, также присваивать значения переменных типа *StringBuilder* в массив строк, то есть из одной строки делать несколько, повторили работу с регулярными выражениями, проверили работу программы с помощью модульных тестов.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программы**

**Program.cs:**

using LibraryForStrings;

using System.Text;

using static System.Net.Mime.MediaTypeNames;

using System.Text.RegularExpressions;

namespace String

{

public class Program

{

static void Main(string[] args)

{

StringArr str = new StringArr();

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

while (isRun) // Консольное меню.

{

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

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

Console.WriteLine("| 1 - Ввод IP-адресов с |");

Console.WriteLine("| разделением их на группы и |");

Console.WriteLine("| выводом групп |");

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

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

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

{

case 1: // Вводим строку и преобразовываем её в структурированные массивы строк

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

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

Console.WriteLine("\nВведите записи IP-адресов:\n");

StringBuilder sb = new StringBuilder();

while (i < n)

{

string input = Console.ReadLine();

bool isIp = true;

if (ValidateInput(input))

{

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

int k = 0;

while (k < 4)

{

if (Convert.ToInt32(number[k]) > 255 || Convert.ToInt32(number[k]) < 0)

{

isIp = false;

}

k++;

}

if (isIp)

{

string ipAdress = input + " ";

sb.Append(ipAdress);

i++;

}

else

{

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

}

}

else

{

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

}

}

str = StringArr.SplitAdress(sb);

Console.WriteLine("\nГруппа A:");

for (int j = 0; j < str.NumberA; j++) { Console.WriteLine(str.GroupA[j]); }

Console.WriteLine("\nГруппа B:");

for (int j = 0; j < str.NumberB; j++) { Console.WriteLine(str.GroupB[j]); }

Console.WriteLine("\nГруппа C:");

for (int j = 0; j < str.NumberC; j++) { Console.WriteLine(str.GroupC[j]); }

Console.WriteLine("\nГруппа D:");

for (int j = 0; j < str.NumberD; j++) { Console.WriteLine(str.GroupD[j]); }

Console.WriteLine("\nГруппа E:");

for (int j = 0; j < str.NumberE; j++) { Console.WriteLine(str.GroupE[j]); }

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+\.\d+\.\d+$"; // Формат: XXX.XXX.XXX.XXX

return Regex.IsMatch(input, pattern);

}

}

}

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Текста библиотеки**

**StringArr.cs:**

using System.Text;

namespace LibraryForStrings

{

public class StringArr

{

public string[] GroupA = new string[1000];

public int NumberA = 0;

public string[] GroupB = new string[1000];

public int NumberB = 0;

public string[] GroupC = new string[1000];

public int NumberC = 0;

public string[] GroupD = new string[1000];

public int NumberD = 0;

public string[] GroupE = new string[1000];

public int NumberE = 0;

/// <summary>

/// Метод для структурирования строки типа StringBuilder в массив строк по классам IP-адресов.

/// </summary>

/// <param name="sb"></param>

/// <returns></returns>

public static StringArr SplitAdress(StringBuilder sb)

{

StringArr stringArr = new StringArr();

string[] ipAdresses = sb.ToString().Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

int i = 0;

while (i < ipAdresses.Length)

{

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

if (Convert.ToInt32(ipAdress[0]) >= 0 && Convert.ToInt32(ipAdress[0]) <= 127)

{

stringArr.GroupA[stringArr.NumberA] = ipAdresses[i];

stringArr.NumberA++;

}

if (Convert.ToInt32(ipAdress[0]) >= 128 && Convert.ToInt32(ipAdress[0]) <= 191)

{

stringArr.GroupB[stringArr.NumberB] = ipAdresses[i];

stringArr.NumberB++;

}

if (Convert.ToInt32(ipAdress[0]) >= 192 && Convert.ToInt32(ipAdress[0]) <= 223)

{

stringArr.GroupC[stringArr.NumberC] = ipAdresses[i];

stringArr.NumberC++;

}

if (Convert.ToInt32(ipAdress[0]) >= 224 && Convert.ToInt32(ipAdress[0]) <= 239)

{

stringArr.GroupD[stringArr.NumberD] = ipAdresses[i];

stringArr.NumberD++;

}

if (Convert.ToInt32(ipAdress[0]) >= 240 && Convert.ToInt32(ipAdress[0]) <= 247)

{

stringArr.GroupE[stringArr.NumberE] = ipAdresses[i];

stringArr.NumberE++;

}

i++;

}

return stringArr;

}

}

}

**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);

try

{

int j = 1;

string numb = "";

while (j < num[0].Length)

{

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

j++;

}

long.Parse(numb);

long.Parse(num[2]);

j = 0;

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

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

if (num[0].Length > 8 && Convert.ToInt32(hour) < 24 && Convert.ToInt32(minutes) < 60)

{

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++;

}

catch (FormatException)

{

}

}

return sb;

}

}

}

**ПРИЛОЖЕНИЕ В**

(обязательное)

**Текст тестов**

**Tests.cs:**

using LibraryForStrings;

using System.Text;

namespace TestForStrings

{

[TestClass]

public class Tests

{

/// <summary>

/// Метод для сравнения массива строк.

/// </summary>

/// <param name="a"></param>

/// <param name="b"></param>

/// <returns></returns>

public static bool CompareString(string[] a, string[] b)

{

if(a.Length != b.Length)

{

return false;

}

int i = 0;

while(i < a.Length)

{

if(a[i] != b[i])

{

return false;

}

i++;

}

return true;

}

/// <summary>

/// Тест при IP-адресе состоящим из нулей.

/// </summary>

[TestMethod]

public void WhenIpAllZero()

{

StringBuilder sb = new StringBuilder();

sb.Append("0.0.0.0");

StringArr result = new StringArr();

result = StringArr.SplitAdress(sb);

string[] str = new string[1000];

str[0] = "0.0.0.0";

Assert.IsTrue(CompareString(str,result.GroupA));

}

/// <summary>

/// Тест при IP-адресе с первым числом меньше чем 128.

/// </summary>

[TestMethod]

public void WhenIpMoreZeroLess128()

{

StringBuilder sb = new StringBuilder();

sb.Append("29.29.29.29");

StringArr result = new StringArr();

result = StringArr.SplitAdress(sb);

string[] str = new string[1000];

str[0] = "29.29.29.29";

Assert.IsTrue(CompareString(str, result.GroupA));

}

/// <summary>

/// Тест при IP-адресе с первым числом меньше 192, но больше 128.

/// </summary>

[TestMethod]

public void WhenIpMore128Less192()

{

StringBuilder sb = new StringBuilder();

sb.Append("150.150.150.0");

StringArr result = new StringArr();

result = StringArr.SplitAdress(sb);

string[] str = new string[1000];

str[0] = "150.150.150.0";

Assert.IsTrue(CompareString(str, result.GroupB));

}

/// <summary>

/// Тест при IP-адресе с первым числом меньше 224, но больше 192.

/// </summary>

[TestMethod]

public void WhenIpMore192Less224()

{

StringBuilder sb = new StringBuilder();

sb.Append("200.200.210.0");

StringArr result = new StringArr();

result = StringArr.SplitAdress(sb);

string[] str = new string[1000];

str[0] = "200.200.210.0";

Assert.IsTrue(CompareString(str, result.GroupC));

}

/// <summary>

/// Тест при IP-адресе с первым числом меньше 240, но больше 224.

/// </summary>

[TestMethod]

public void WhenIpMore224Less240()

{

StringBuilder sb = new StringBuilder();

sb.Append("230.230.0.0");

StringArr result = new StringArr();

result = StringArr.SplitAdress(sb);

string[] str = new string[1000];

str[0] = "230.230.0.0";

Assert.IsTrue(CompareString(str, result.GroupD));

}

/// <summary>

/// Тест при IP-адресе с первым числом меньше 247, но больше 240.

/// </summary>

[TestMethod]

public void WhenIpMore240Less247()

{

StringBuilder sb = new StringBuilder();

sb.Append("241.0.0.0");

StringArr result = new StringArr();

result = StringArr.SplitAdress(sb);

string[] str = new string[1000];

str[0] = "241.0.0.0";

Assert.IsTrue(CompareString(str, result.GroupE));

}

/// <summary>

/// Тест при IP-адресе не принадлежащему не одному классу.

/// </summary>

[TestMethod]

public void WhenIpMore247Less255()

{

StringBuilder sb = new StringBuilder();

sb.Append("250.0.0.0");

StringArr result = new StringArr();

result = StringArr.SplitAdress(sb);

string[] str = new string[1000];

str[0] = "250.0.0.0";

Assert.IsFalse(CompareString(str, result.GroupA) || CompareString(str, result.GroupB) || CompareString(str, result.GroupC) || CompareString(str, result.GroupD) || CompareString(str, result.GroupE));

}

/// <summary>

/// Тест при нескольких IP-адресов различных классов.

/// </summary>

[TestMethod]

public void WhenAllGroupHaveIp()

{

StringBuilder sb = new StringBuilder();

sb.Append("0.0.0.0 128.0.0.0 192.0.0.0 224.0.0.0 240.0.0.0");

StringArr result = new StringArr();

result = StringArr.SplitAdress(sb);

string[] str1 = new string[1000];

str1[0] = "0.0.0.0";

string[] str2 = new string[1000];

str2[0] = "128.0.0.0";

string[] str3 = new string[1000];

str3[0] = "192.0.0.0";

string[] str4 = new string[1000];

str4[0] = "224.0.0.0";

string[] str5 = new string[1000];

str5[0] = "240.0.0.0";

Assert.IsTrue(CompareString(str1, result.GroupA) && CompareString(str2, result.GroupB) && CompareString(str3, result.GroupC) && CompareString(str4, result.GroupD) && CompareString(str5, result.GroupE));

}

/// <summary>

/// Тест при нескольких количествах IP-адресов одного класса.

/// </summary>

[TestMethod]

public void WhenGroupHaveManyIp()

{

StringBuilder sb = new StringBuilder();

sb.Append("0.0.0.0 0.20.0.0 30.0.0.0 127.0.0.0 127.255.255.255");

StringArr result = new StringArr();

result = StringArr.SplitAdress(sb);

string[] str = new string[1000];

str[0] = "0.0.0.0";

str[1] = "0.20.0.0";

str[2] = "30.0.0.0";

str[3] = "127.0.0.0";

str[4] = "127.255.255.255";

Assert.IsTrue(CompareString(str, result.GroupA));

}

}

}