Министерство науки и высшего образованияРоссийской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет

имени В.Ф.Уткина»

Кафедра «Электронные вычислительные машины»

Отчет по лабораторной работе №4

на тему

«Регулярные выражения»

по дисциплине  
**«Визуальное программирование»**

Выполнили:

Студенты группы №140

Бригада 5

Сафаров Д. А.

Тимохин Е. С.

Проверили:  
ст. преп. Хизриева Н.И.

ст. преп. Бастрычкин А.С.

**Цель работы**

Изучить работу с регулярными выражениями на языке C#.

**Задание**

1) Написать метод, который определяет количество входящих в заданную строку почтовых индексов (почтовый индекс состоит из 6 цифр).

2) Дана строка — предложение на русском языке. Поменять местами первую и последнюю буквы каждого слова.

3) Дана строка, содержащая помимо прочей информации номера телефонов в федеральном формате. Скрыть все цифры городской части номера кроме самой первой под символами x. Считать, что код города может содержать от трёх до пяти цифр, а городской номер — от 7 до 5 цифр соответственно.

4) Выяснить, какими могут быть российские автомобильные номера  
(с кодом региона), составить соответствующее регулярное выражение и написать метод, который находит в строке все автомобильные номера и возвращает их в виде последовательности.

5) Дана строка. Сохранить в новую строку все содержащиеся в ней   
IPv4-адреса в десятичной записи с точками через разделитель.

**Практическая часть**

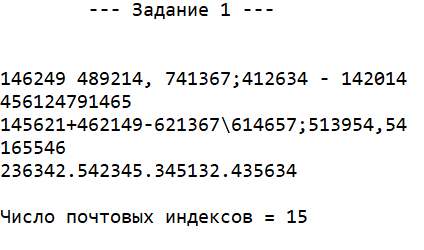
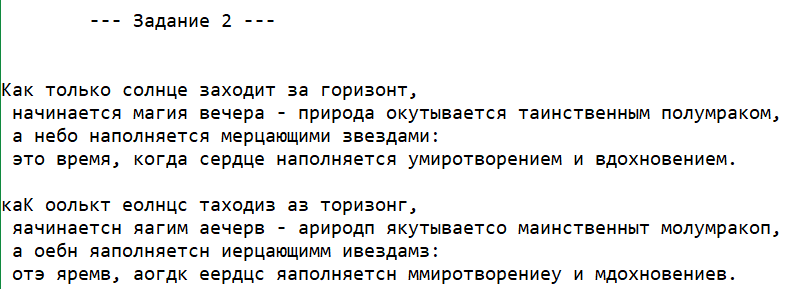
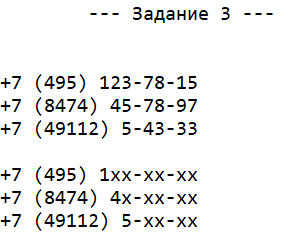
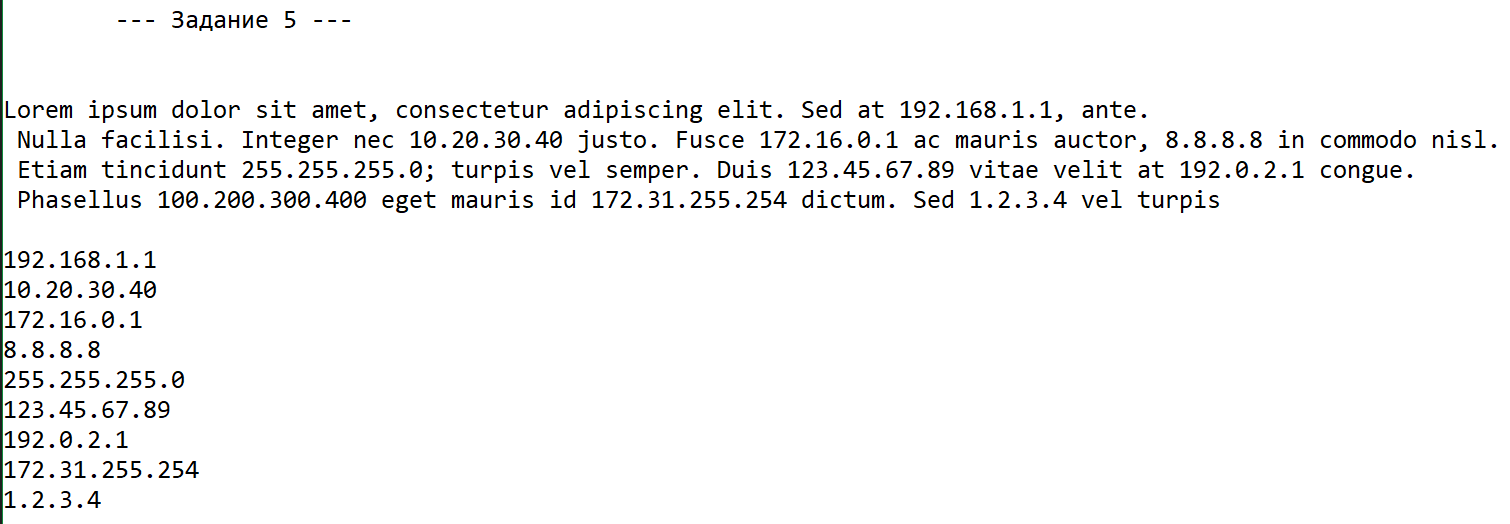
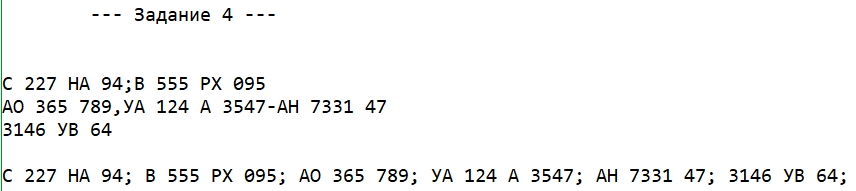
Код программы представлен в приложении А. Результат её работы представлен на рисунках 1-2.

Рисунок 1 – Результаты работы программы, задания 1-3

Рисунок 2 – Результаты работы программы, задания 4-5



Приложение А. Текст программы

**Regulars.cs**

using System.Text;

using System.Text.RegularExpressions;

namespace RegexLabLogic

{

public static class Regulars

{

// Подсчёт входящих почтовых индексов (с любыми разделителями)

public static int CountPostcodes(this string str)

{

return Regex.Count(str, @"\b\d{6}\b");

}

// Меняет местами первую и последнюю букву каждого слова

public static string WordToDorw(this string sentence)

{

var matches = Regex.Matches(sentence, @"\w+").ToArray(); // Получить все

слова

var words = new string[matches.Length]; // Массив слов

var dorws = new string[matches.Length]; // Массив обработанных слов

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

{

words[i] = matches.GetValue(i).ToString(); // Получить слово

StringBuilder dorwSB = new(words[i]);

dorwSB[0] = words[i][words[i].Length-1]; // Поменять местами первую

dorwSB[words[i].Length-1] = words[i][0]; // и последнюю буквы

dorws[i] = dorwSB.ToString(); // Получить обработанное слово

}

StringBuilder newSentenceSB = new(sentence); // Изменённое предложение

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

{

newSentenceSB.Replace(words[i], dorws[i]); // Замена слов

}

return newSentenceSB.ToString();

}

// Скрывает городскую часть номера со второй цифры

public static string HidePhonePart(this string str)

{

// Регулярное выражение для телефонов

string phonePattern =

@"\+7 ((\(\d{3}\) \d{3})|(\(\d{4}\) \d{2})|(\(\d{5}\) \d{1}))-\d{2}-\d{2}";

// Строка со скрытыми номерами

StringBuilder hiddenSB = new(str);

// Количество номеров

int phonesCount = Regex.Count(str, phonePattern);

for (int k = 0; k < phonesCount; k++)

{

// Взять телефон

StringBuilder hiddenPhoneSB = new(Regex.Match(hiddenSB.ToString(),

phonePattern).Value);

// Проход с конца

for (int i = hiddenPhoneSB.Length - 1; i >= 0; i--)

{

// Скрывать цифры

if (hiddenPhoneSB[i] != '-')

hiddenPhoneSB[i] = 'x';

// До первой цифры после пробела исключительно

if (hiddenPhoneSB[i-2] == ' ')

break;

}

// Замена телефона на скрытый

hiddenSB.Replace(Regex.Match(hiddenSB.ToString(), phonePattern).Value,

hiddenPhoneSB.ToString());

}

return hiddenSB.ToString();

}

// Возвращает входящие в строку автономера в виде последовательности с

// раделителем ';'

public static string GetLicensePlates(this string str)

{

// Регулярное выражение для автономеров

string platePattern =

@"(([АВЕКМНОРСТУХ]{2} \d{3,4})|

([АВЕКМНОРСТУХ]{2} \d{3,4} [АВЕКМНОРСТУХ]{1})|

([АВЕКМНОРСТУХ]{2} \d{2} [АВЕКМНОРСТУХ]{2})|

([АВЕКМНОРСТУХ]{1} ((\d{3} [АВЕКМНОРСТУХ]{2})|(\d{4} )))|

(\d{3,4} [АВЕКМНОРСТУХ]{1,2})) \d{2,4}";

StringBuilder platesSB = new();

var plates = Regex.Matches(str, platePattern).ToArray();

foreach (var adress in adresses)

adressesSB.Append(adress.Value + separator);

return platesSB.ToString();

}

// Возвращает строку, содержащую IPv4 адреса в десятичном формате из оригинальной строки,

// разделённые указанным разделителем

public static string GetIPv4Adresses(this string str, string separator = "; ")

{

// Регулярное выражение для IPv4

string adressPatern = @"\b(((2(([0-4][0-9])|5[0-5])|(1[0-9][0-9])|([1-9][0-9])|[0-9])\.){3}(2(([0-4][0-9])|5[0-5])|(1[0-9][0-9])|([1-9][0-9])|[0-9]))\b";

StringBuilder adressesSB = new();

var adresses = Regex.Matches(str, adressPatern).ToArray();

foreach (var adress in adresses)

{

adressesSB.Append(adress.Value + separator);

}

return adressesSB.ToString();

}

}

}

**Program.cs**

using RegexLabLogic;

#region 1

Console.WriteLine("\t--- Задание 1 ---\n\n");

string postcodes = "146249 489214, 741367;412634 - 142014\r\n456124791465\r\n145621+462149-621367\\614657;513954,54\r\n165546\r\n236342.542345.345132.435634";

Console.WriteLine($"{postcodes}\n\n

Число почтовых индексов = {postcodes.CountPostcodes()}\n\n");

#endregion

#region 2

Console.WriteLine("\t--- Задание 2 ---\n\n");

string sentence = "Как только солнце заходит за горизонт,\n" +

" начинается магия вечера - природа окутывается таинственным полумраком,\n" +

" а небо наполняется мерцающими звездами:\n" +

" это время, когда сердце наполняется умиротворением и вдохновением.";

Console.WriteLine($"{sentence}\n\n{sentence.WordToDorw()}\n\n");

#endregion

#region 3

Console.WriteLine("\t--- Задание 3 ---\n\n");

string phones = "+7 (495) 123-78-15\n+7 (8474) 45-78-97\n+7 (49112) 5-43-33";

Console.WriteLine($"{phones}\n\n{phones.HidePhonePart()}\n\n");

#endregion

#region 4

Console.WriteLine("\t--- Задание 4 ---\n\n");

string licensePlates = "С 227 НА 94;В 555 РХ 095\nАО 365 789,УА 124 А 3547-АН 7331 47\n3146 УВ 64";

Console.WriteLine($"{licensePlates}\n\n{licensePlates.GetLicensePlates()}\n\n");

#endregion

#region 5

Console.WriteLine("\t--- Задание 5 ---\n\n");

// Здесь 9 адресов и один неправильный

string textWithAdresses =

"Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed at 192.168.1.1, ante.\n" +

" Nulla facilisi. Integer nec 10.20.30.40 justo. Fusce 172.16.0.1 ac mauris auctor, 8.8.8.8 in commodo nisl.\n" +

" Etiam tincidunt 255.255.255.0; turpis vel semper. Duis 123.45.67.89 vitae velit at 192.0.2.1 congue.\n" +

" Phasellus 100.200.300.400 eget mauris id 172.31.255.254 dictum. Sed 1.2.3.4 vel turpis";

Console.WriteLine($"{textWithAdresses}\n\n

{textWithAdresses.GetIPv4Adresses("\n")}\n\n");

#endregion