МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Гомельский государственный технический

университет имени П. О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Лабораторная работа №4

По дисциплине «Объектно-ориентированное программирование»

**«Обработка строковых данных. Регулярные выражения»**

Выполнил: студент группы ИТИ-21

Говядкова П. Ю.

Принял: преподаватель

Карабчикова Е. А.

Гомель 2020

**Цель работы:** изучить основы синтаксиса объектно-ориентированного языка программирования, реализацию свойств, методов класса.

**Задание:**

1. Разработать приложения, решающие поставленную задачу (Рисунок 1 и 2).
2. Разработать не менее 15 модульных тестов для тестирования.



Рисунок 1 – Вариант 1-ого задания



Рисунок 2 – Вариант 2-ого задания

На Рисунке 3 изображена структура решения.

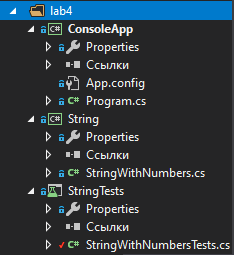


Рисунок 3 – Структура решения

На Рисунке 4 изображено начало выполнения программы.

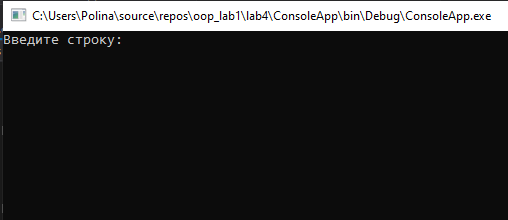


Рисунок 4 – Начало выполнения программы, ожидания ввода с клавиатуры

Выполнение 1-ого пункта: ввода строки и поиска арифметические выражения вида a ® b, где ® — один из знаков +, -, \*, /. Вывод и вычисление значений.

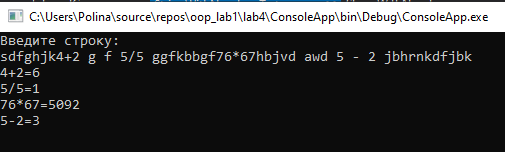


Рисунок 5 – Выполнение 1-ого пункта: ввода строки и поиска арифметических выражений.

Выполнение 2-ого пункта: задан текст на русском языке. Сформировать массив из предложений следующего вида: Согласная буква “в” встречается столько-то раз, и т. д.

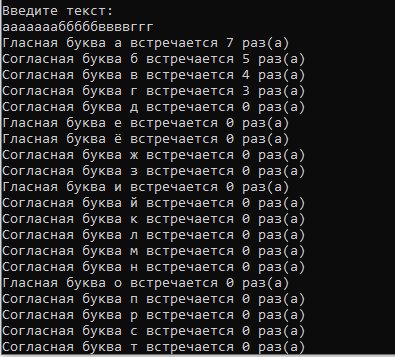


Рисунок 6 – Выполнение 2-ого пункта: задан текст на русском языке и формирование массива из предложений

На Рисунках 7 – 8 изображены результаты выполнения тестов:

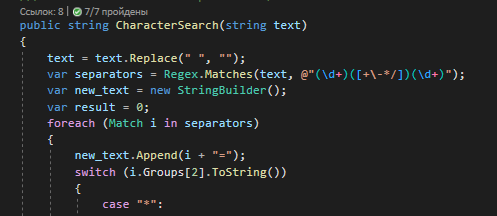


Рисунок 7 – Результат успешного выполнения теста.

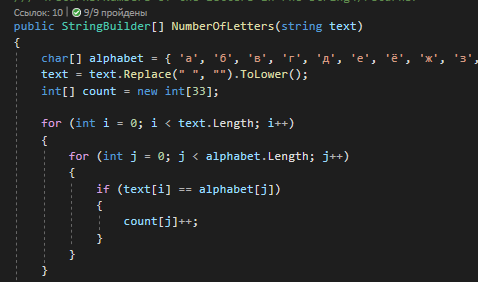


Рисунок 8 – Результат успешного выполнения теста.

**Вывод:** изучены основы синтаксиса объектно-ориентированного языка программирования. Реализован класс для работы со строками и регулярными выражениями.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

Program

using String;

using System;

using System.Text;

namespace ConsoleApp

{

/// <summary>

/// Program

/// </summary>

internal class Program

{

/// <summary>

/// Defines the entry point of the application.

/// </summary>

/// <param name="args">The arguments.</param>

private static void Main(string[] args)

{

StringBuilder[] array;

Console.WriteLine("Введите строку: ");

string text = Console.ReadLine();

StringWithNumbers Text = new StringWithNumbers(text);

Console.WriteLine($"{Text.CharacterSearch(text)}");

Console.WriteLine("\nВведите текст: ");

string new\_text;

new\_text = Console.ReadLine();

array = Text.NumberOfLetters(new\_text);

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

{

Console.Write(array[i]);

Console.Write('\n');

}

Console.ReadKey();

}

}

}

StringWithNumbers

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Text.RegularExpressions;

namespace String

{

/// <summary>

/// StringWithNumbers

/// </summary>

public class StringWithNumbers

{

/// <summary>

/// The text

/// </summary>

private string \_text;

/// <summary>

/// Initializes a new instance of the <see cref="StringWithNumbers"/> class.

/// </summary>

/// <param name="text">The text.</param>

public StringWithNumbers(string text)

{

this.\_text = text;

}

/// <summary>

/// Search for characters in a string.

/// </summary>

/// <param name="text">The text.</param>

/// <returns>Arithmetic expressions</returns>

public string CharacterSearch(string text)

{

text = text.Replace(" ", "");

var separators = Regex.Matches(text, @"(\d+)([+\-\*/])(\d+)");

var new\_text = new StringBuilder();

var result = 0;

foreach (Match i in separators)

{

new\_text.Append(i + "=");

switch (i.Groups[2].ToString())

{

case "\*":

{

result = int.Parse(i.Groups[1].ToString()) \* int.Parse(i.Groups[3].ToString());

break;

}

case "/":

{

result = int.Parse(i.Groups[1].ToString()) / int.Parse(i.Groups[3].ToString());

break;

}

case "+":

{

result = int.Parse(i.Groups[1].ToString()) + int.Parse(i.Groups[3].ToString());

break;

}

case "-":

{

result = int.Parse(i.Groups[1].ToString()) - int.Parse(i.Groups[3].ToString());

break;

}

}

new\_text.Append(result+"\n");

}

return new\_text.ToString();

}

/// <summary>

/// Numbers of the letters.

/// </summary>

/// <param name="text">The text.</param>

/// <returns>Numbers of the letters in rhe string</returns>

public StringBuilder[] NumberOfLetters(string text)

{

char[] alphabet = { 'а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я' };

text = text.Replace(" ", "").ToLower();

int[] count = new int[33];

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

{

for (int j = 0; j < alphabet.Length; j++)

{

if (text[i] == alphabet[j])

{

count[j]++;

}

}

}

StringBuilder[] array = new StringBuilder[alphabet.Length];

for (int j = 0, i = 0; j < alphabet.Length; j++, i++)

{

if (alphabet[j] == 'а' || alphabet[j] == 'о' || alphabet[j] == 'у' || alphabet[j] == 'ы' || alphabet[j] == 'э' || alphabet[j] == 'я' || alphabet[j] == 'ё' || alphabet[j] == 'е' || alphabet[j] == 'ю' || alphabet[j] == 'и')

{

array[i] = new StringBuilder("Гласная буква " + alphabet[j] + " встречается " + Convert.ToString(count[j]) + " раз(а) ");

}

else array[i] = new StringBuilder("Согласная буква " + alphabet[j] + " встречается " + Convert.ToString(count[j]) + " раз(а) ");

}

return array;

}

}

}