|  |  |
| --- | --- |
|  | /storage/emulated/0/.polarisOffice5/polarisTemp/image1.png |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждениевысшего профессионального образования «Московский технологический университет»  МИРЭА | | |

Институт Информационных Технологий

Кафедра Корпоративных Информационных Систем

**ОТЧЕТ**

по Лабораторной Работе № 6

на тему

«Тип данных — строка»

по дисциплине

«Процедурное Программирование»

Выполнил студент группы ИКБО-07-17 Акжигитов Р. Р.

Принял асс. каф. КИС Алдобаева В. Н.

Выполнено «21» ноября 2017 г.

Зачтено «\_\_\_» \_\_\_\_\_\_\_\_\_ 2017 г.

Москва, 2017

**Задание**

Обработать введенную пользователем строку в соответствии с индивидуальным заданием. Каждое задание обработки должно осуществляться при нажатии соответствующей кнопки. Вывод должен производиться с возможностью копирования и без потери введенной пользователем строки.

**Вариант №1**

1. Подсчитать количество символов, не являющихся буквами
2. Заменить все запятые на точку с запятой
3. Подсчитать сумму всех чисел в строке
4. Заменить каждую четную букву ‘b’ на ‘б’

**Теоретическая часть**

Строка – некоторая конечная последовательность символов. Возможно рассматривать строку как массив символов, но подход не учитывает некоторые особенности строк:

1. Значения строк часто используются повторно без изменений.
2. Возможный диапазон значений элемента строки (символа) невелик.
3. Необходимость перестановки отдельных элементов (символов) в строке возникает исключительно редко (смысл при сортировке строки по коду символов, например, теряется)
4. Операция сложения для строк – конкатенация.

Во многих языках программирования строки выносят в отдельный тип данных со специальной обработкой.

К символам в строке можно обращаться как к элементам массива.

Все строки в C# по умолчанию представлены в кодировке UNICODE UTF-16. В данной кодировке не все символы занимают одно знакоместо, например, буква «Ё» может быть представлена как буква «Е» с диактрическим знаком «точки над символом».

Для строк определены следующие основные операции: конкатенация строк, чтение элемента, операции сравнения ==, !=.

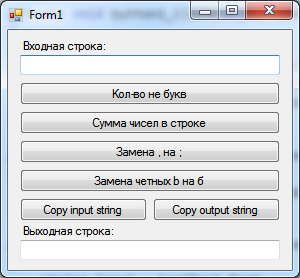
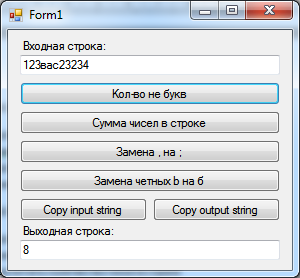
Тип String не изменяем, но StringBuilder изменяем. Тем более он содержит методы для манипуляций с отдельными элементами строк.

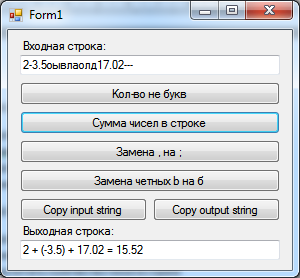
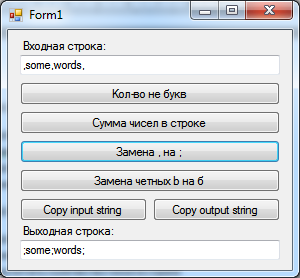
В типе StringBuilder определены следующие методы: очистка буфера строки (стирание всех символов и установка длины в 0), вставка текстового представления элемента в строке, удаление фрагмента заданной длины, замена символа (фрагмента) на другой, так же можно по тому или иному признаку выделить подстроку из строки (необходимо запомнить начало подстроки).

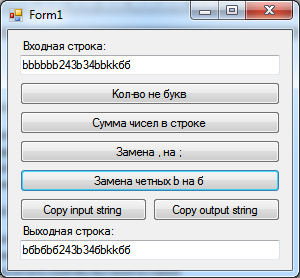
**Алгоритм решения задачи**

1. Разместить необходимые компоненты на форме (поля для ввода и вывода, кнопки триггеры для исполнения функций).
2. Подсчитать кол-во символов, не являющихся буквами, можно с помощью метода **Char.IsLetter**.
3. Заменить запятую на точку можно с помощью **String.Replace(oldChar, newChar);**
4. Подсчитать сумму всех чисел в строке. Если стоит знак «минус» и следующий элемент «цифра», то число отрицательное. Считывать цифры поэлементно до тех пор, пока очередной элемент не станет не «цифрой», далее можно преобразовать строку из цифр в число. Записывая числа в массив, можем найти сумму чисел, линейно проходя через все элементы и складывая их.
5. Замена каждой четной «b» на «б». Цикл по строке. Если символ - английская «б», то +1 в счетчик, если счетчик четен, то заменяем букву с этим индексом на русскую, либо создаем новую строку, и замену делаем уже в ней, если необходима, если нет, то просто копируем посимвольно.

**Тестирование**



**Заключение**

В ходе выполнения лабораторной работы я узнал много нового про строки, как их обрабатывать, какие основные операции существуют с ними, чем они отличаются от массива символов.

**Исходный код**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Text.RegularExpressions;

namespace laba6

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string input = textBox1.Text;

Regex regex = new Regex(@"(\W|\d)");

MatchCollection matches = regex.Matches(input);

textBox2.Text = matches.Count.ToString();

}

private void button2\_Click(object sender, EventArgs e)

{

textBox2.Text = textBox1.Text.Replace(",", ";");

}

private void button3\_Click(object sender, EventArgs e)

{

string input = textBox1.Text;

Regex regex = new Regex(@"-?\d+(\.\d+)?");

MatchCollection matches = regex.Matches(input);

List<String> matchesInString = new List<String>();

foreach (Match match in matches)

{

matchesInString.Add(match.Value);

}

textBox2.Text = String.Join(" + ", matchesInString.Select(x => (x[0] == '-') ? String.Format("({0})", x) : x)) + " = " + matchesInString.Select(x => double.Parse(x.Replace(".", ","))).Aggregate((a, b) => a + b).ToString().Replace(",", ".");

}

private void button4\_Click(object sender, EventArgs e)

{

string input = textBox1.Text;

List<int> bIndices = input.Select((x, ind) => (x == 'b') ? ind : -1).Where(x => x >= 0).ToList();

var evenIndices = bIndices.Where((x, ind) => ind % 2 == 1);

var total = input.Select((y, inx) => (Array.IndexOf(evenIndices.ToArray(), inx) >= 0) ? 'б' : y);

textBox2.Text = String.Join("", total);

}

private void button5\_Click(object sender, EventArgs e)

{

Clipboard.SetText(textBox1.Text);

}

private void button6\_Click(object sender, EventArgs e)

{

Clipboard.SetText(textBox2.Text);

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

}

}