Московский государственный технический университет
им. Н.Э. Баумана

Кафедра
“Системы обработки информации и управления”
(ИУ – 5)

Лабораторная работа по №5 по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:
студент гр. ИУ5 - 31
Саадуев А.С.

“27” декабря 2017 г.

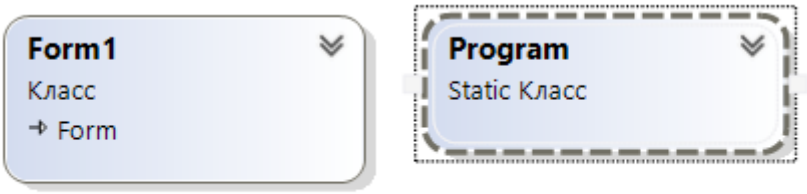
Москва – 2017 г.

Задание:

Разработать программу, реализующую вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера.

1. Программа должна быть разработана в виде библиотеки классов на языке C#.
2. Использовать самый простой вариант алгоритма без оптимизации.
3. Дополнительно возможно реализовать вычисление расстояния Дамерау-Левенштейна (с учетом перестановок соседних символов).
4. Модифицировать предыдущую лабораторную работу, вместо поиска подстроки используется вычисление расстояния Левенштейна.
5. Предусмотреть отдельное поле ввода для максимального расстояния. Если расстояние Левенштейна между двумя строками больше максимального, то строки считаются несовпадающими и не выводятся в список результатов.

Диаграмма классов:



Код:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace _4a
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
```

```
        }

        List<string> list = new List<string>();
        string[] textArray;
        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            OpenFileDialog fd = new OpenFileDialog();
            fd.Filter = "текстовые файлы|*.txt";
            if (fd.ShowDialog() == DialogResult.OK)
            {
                Stopwatch t = new Stopwatch();
                t.Start();
                //Чтение файла в виде строки
                string text = File.ReadAllText(fd.FileName);
                char[] separators = new char[] { ' ', '.', ',', '!', '?', '/', '\t', '\n'
};
                textArray = text.Split(separators);

                foreach (string strTemp in textArray)
                {
                    //Удаление пробелов в начале и конце строки
                    string str = strTemp.Trim();
                    //Добавление строки в список, если строка не содержится в списке
                    if (!list.Contains(str)) list.Add(str);
                }

                t.Stop();
                this.textBox2.Text = t.Elapsed.ToString();
                string a = string.Join(" ", list.ToArray());
                textBox1.Text = a.ToString();
            }
            else
            {
                MessageBox.Show("Необходимо выбрать файл");
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string word = this.textBox3.Text.Trim();
            List<string> tempList = new List<string>();
            Stopwatch t = new Stopwatch();
            t.Start();
            foreach (string str in list)
            {
                if (str.Contains(word))
                {
                    tempList.Add(str);
                }
            }
            t.Stop();
           textBox4.Text = t.Elapsed.ToString();
            listBox1.BeginUpdate();
            listBox1.Items.Clear();

//Вывод результатов поиска
foreach (string str in tempList)
            {
               listBox1.Items.Add(str);
            }
```

```
            listBox1.EndUpdate();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            string word1 = this.textBox5.Text.Trim();
            //textBox6.Text = word1.ToString();



            //listBox2.BeginUpdate();
            listBox2.Items.Clear();
            //listBox3.BeginUpdate();
            listBox3.Items.Clear();
            int b = 0;
           int c = int.Parse(textBox6.Text);
            foreach (string strTemp in textArray)
            {
                //textBox7.Text=Distance(word1, strTemp).ToString();
                //Добавление строки в список, если строка не содержится в списке
                b = Distance(word1, strTemp);
                listBox2.Items.Add(b.ToString());
                if(b<=c) listBox3.Items.Add(strTemp);

            }

        }
    public static int Distance(string str1Param, string str2Param)
        {
            if ((str1Param == null) || (str2Param == null)) return -1;
            int str1Len = str1Param.Length;
            int str2Len = str2Param.Length;
            //Если хотя бы одна строка пустая, возвращается длина другой строки
            if ((str1Len == 0) && (str2Len == 0)) return 0;
            if (str1Len == 0) return str2Len;
            if (str2Len == 0) return str1Len;
            //Приведение строк к верхнему регистру
            string str1 = str1Param.ToUpper();
            string str2 = str2Param.ToUpper();
            //Объявление матрицы
            int[,] matrix = new int[str1Len + 1, str2Len + 1];
            //Инициализация нулевой строки и нулевого столбца матрицы
            for (int i = 0; i <= str1Len; i++) matrix[i, 0] = i;
            for (int j = 0; j <= str2Len; j++) matrix[0, j] = j;

//Вычисление расстояния Дамерау-Левенштейна
for (int i = 1; i <= str1Len; i++)
            {
                for (int j = 1; j <= str2Len; j++)
                {
                    //Эквивалентность символов, переменная symbEqual соответствует
                    //m(s1[i], s2[j])
                int symbEqual = ((str1.Substring(i - 1, 1) == str2.Substring(j
                - 1, 1)) ? 0 : 1);
                    int ins = matrix[i, j - 1] + 1; //Добавление
                    int del = matrix[i - 1, j] + 1; //Удаление
                    int subst = matrix[i - 1, j - 1] + symbEqual; //Замена
                                                                  //Элемент матрицы
вычисляется как минимальный из трех случаев
                    matrix[i, j] = Math.Min(Math.Min(ins, del), subst);
                    //Дополнение Дамерау по перестановке соседних символов
                    if ((i > 1) && (j > 1) &&
                    (str1.Substring(i - 1, 1) == str2.Substring(j - 2, 1)) &&
                    (str1.Substring(i - 2, 1) == str2.Substring(j - 1, 1)))
                    {
```

```
                    matrix[i, j] = Math.Min(matrix[i, j], matrix[i - 2, j - 2]
                    + symbEqual);
                }
            }
        }
        //Возвращается нижний правый элемент матрицы
        return matrix[str1Len, str2Len];
    }


}
}
```

Скриншот: