Лабораторная работа № 5 по курсу
“Базовые компоненты интернет-технологий”

Лазарев Станислав Алексеевич
РТ5-31
МГТУ им. Баумана

Описание задания лабораторной работы.

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.

Код программы:

```
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 WindowsFormsApplication4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        List<string> list = new List<string>();
        private void button1_Click(object sender, EventArgs e)
        {

            OpenFileDialog fn = new OpenFileDialog();
            fn.Filter = "текстовые файлы |*.txt";
            if (fn.ShowDialog() == DialogResult.OK)
            {
                Stopwatch timer = new Stopwatch();
                timer.Start();
                string text = File.ReadAllText(fn.FileName);
```

```
                char[] separators = new char[] { ' ', '.', ',', '!', '?', '/', '\t', '\n' };
                string[] textArray = text.Split(separators);
                foreach (string strTemp in textArray)
                {
                    string str = strTemp.Trim();
                    if (!list.Contains(str)) list.Add(str);

                }
                timer.Stop();

                this.textBox1.Text = timer.Elapsed.ToString();
                this.textBox2.Text = list.Count.ToString();
            }
            else
            {
                MessageBox.Show("Необхоимо выбрать файл");
            }
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            string word = this.textBox3.Text.Trim();
            if (!string.IsNullOrWhiteSpace(word) && list.Count > 0)
            {
                int maxDist;
                if (!int.TryParse(this.textBox4.Text.Trim(), out maxDist))
                {
                    {
                        MessageBox.Show("Необходимо указать максимальное расстояние");
                        return;
                    }
                    if (maxDist < 1 || maxDist > 5)
                    {
                        MessageBox.Show("Максимальное расстояние должно быть в диапазоне от
1 до 5");

                        return;
                    }
                }
                string WordUpper = word.ToUpper();
                List<Tuple<string, int>> tempList = new List<Tuple<string, int>>();
                Stopwatch timer = new Stopwatch();
                timer.Start();
                foreach (string str in list)
                {
                    int dist = EditDistance.Distance(str.ToUpper(), WordUpper);
                    if (dist <= maxDist)
                    {
                        tempList.Add(new Tuple<string, int>(str, dist));
                    }

                }
```

```
                timer.Stop();
                this.textBox5.Text = timer.Elapsed.ToString();
                this.listBox1.BeginUpdate();
                this.listBox1.Items.Clear();
                foreach (var x in tempList)
                {
                    string temp = x.Item1 + "(расстояние Левенштайна =" + x.Item2.ToString()
+ ")";
                    this.listBox1.Items.Add(temp);
                }
                this.listBox1.EndUpdate();


            }
            else
            {

                MessageBox.Show("Необходимо выбрать файл и ввести слово для поиска");
            }
        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void label5_Click(object sender, EventArgs e)
        {

        }
    }
    public static class EditDistance
    {
        public static int Distance(string str1Param, string str2Param)
        {
            if ((str1Param == null) || (str2Param == null)) return -1;
            int str1Len = str1Param.Length;
            int str2Len = str2Param.Length;
            if ((str1Len == 0) && (str2Len == 0)) return 0;
            if (str1Len == 0) return str2Len;
            if (str2Len == 0) return str1Len;
            string str1 = str1Param.ToUpper();
            string str2 = str2Param.ToUpper();
            int[,] matrix = new int[str1Len + 1, str2Len + 1];
            for (int i = 0; i <= str1Len; i++) matrix[i, 0] = i;
            for (int j = 0; j <= str2Len; j++) matrix[0, j] = j;
            for (int i = 1; i <= str1Len; i++)
            {
                for (int j = 1; j <= str2Len; 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
                    matrix[i, j] = Math.Min(Math.Min(ins, del), subst);
                    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];
        }

    }
}
```

Диаграмма классов:

Пример консольного вывода: