|  |  |  |
| --- | --- | --- |
| Защищено:  Ю.Е. Гапанюк    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. |  | Демонстрация:  Ю.Е. Гапанюк    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. |

**Отчет по лабораторной работе № 5 по курсу**

**Базовые компоненты интернет-технологий**

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент гр. ИУ5Ц-52Б  (учится с гр.ИУ5-32Б) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Железов А.С. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. |

**Москва, МГТУ - 2017**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Задание к Лабораторной роботе № 5:

# 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.

**Текст программы:**

**Form1.cs\***

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Diagnostics;

using EditDistanceProject;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

label1.Visible = false;

textBoxFind.Visible = false;

buttonExact.Visible = false;

label2.Visible = false;

label3.Visible = false;

textBoxMaxDist.Visible = false;

buttonApprox.Visible = false;

label4.Visible = false;

listBoxResult.Visible = false;

}

List<string> list = new List<string>();

private void buttonLoadFile\_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();

string text = File.ReadAllText(fd.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) && str != "") list.Add(str);

}

t.Stop();

label1.Visible = true;

label1.Text = "Время чтения файла: " + t.Elapsed.ToString();

textBoxFind.Visible = true;

buttonExact.Visible = true;

label3.Visible = true;

textBoxMaxDist.Visible = true;

buttonApprox.Visible = true;

}

else MessageBox.Show("Необходимо выбрать файл");

}

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

{

string word = textBoxFind.Text.Trim();

if (!string.IsNullOrWhiteSpace(word) && list.Count > 0)

{

string wordUpper = word.ToUpper();

List<string> tempList = new List<string>();

Stopwatch t = new Stopwatch();

t.Start();

foreach (string str in list)

{

if (str.ToUpper().Contains(wordUpper))

{

tempList.Add(str);

}

}

t.Stop();

label2.Visible = true;

listBoxResult.Visible = true;

label2.Text ="Время поиска " + t.Elapsed.ToString();

listBoxResult.BeginUpdate();

listBoxResult.Items.Clear();

int i = 0;

foreach (string str in tempList)

{

i++;

listBoxResult.Items.Add(i + ". " + str);

}

listBoxResult.EndUpdate();

}

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

}

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

{

string word = textBoxFind.Text.Trim();

if (!string.IsNullOrWhiteSpace(word) && list.Count > 0)

{

int maxDist;

if (!int.TryParse(textBoxMaxDist.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 t = new Stopwatch();

t.Start();

foreach (string str in list)

{

int dist = EditDistance.Distance(str.ToUpper(), wordUpper);

if (dist <= maxDist)

{

tempList.Add(new Tuple<string, int>(str, dist));

}

}

t.Stop();

label4.Visible = true;

label4.Text ="MethodWordSearchTime: " + t.Elapsed.ToString() + "; Done";

listBoxResult.Visible = true;

listBoxResult.BeginUpdate();

listBoxResult.Items.Clear();

int i = 0;

foreach (var x in tempList)

{

i++;

string temp =i + ". " + x.Item1 + " (расстояние=" + x.Item2.ToString() +

")";

listBoxResult.Items.Add(temp);

}

listBoxResult.EndUpdate();

}

else

{

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

}

}

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

{

Close();

}

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

{

}

}

}

namespace EditDistanceProject

{

public static class EditDistance

{

/// <summary>

/// Вычисление расстояния Дамерау-Левенштейна

/// </summary>

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];

}

}

}

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

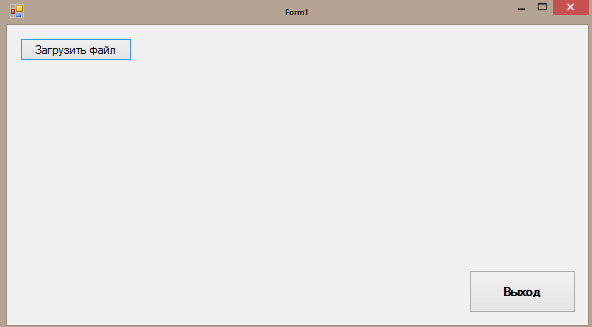
Application.Run(new Form1());

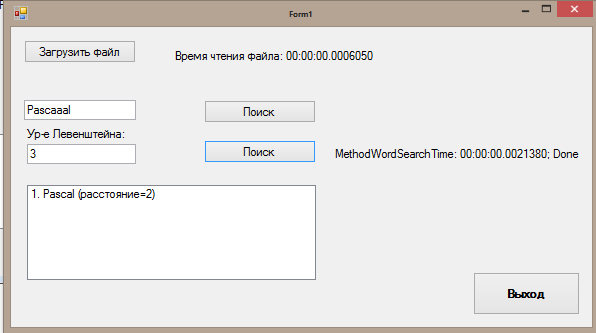
}

}

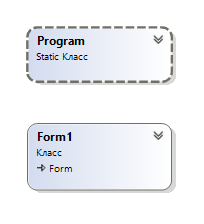
}

**Результаты выполнения программы:**





**Диаграмма классов:**

****