**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**Московский государственный технический университет**

**им. Н.Э. Баумана**

**(МГТУ им. Н.Э. Баумана)**

**Радиотехнический факультет (РТ)**

Лабораторная работа № 5

По дисциплине: «Базовые компоненты интернет технологий»

# Тема: «Разработать программу, реализующую вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера»

Выполнил: Астанов Э.М.,

студент группы РТ5-31

Проверил: Гапанюк Ю.Е.,

Преподаватель каф. ИУ5

г. Москва 2017 г.

Описание задания лабораторной работы:

Разработать программу, реализующую работу с классами.

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:

Файл DamerauLivenstein.cs

namespace DamerauLivensteinLib

{

public static class DamerauLivenstein

{

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);

}

}

}

return matrix[str1Len, str2Len];

}

}

}

Файл Form1.cs:

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

{

//слово из поисковой строки

string word = this.textBoxFind.Text.Trim();

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

{

int maxDist;

if(!int.TryParse(this.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;

}

//Слово для поиска в верхнем регистре

string wordUpper = word.ToUpper();

//Временные результаты поиска

List<Tuple<string, int>> tempList = new List<Tuple<string,int>>();

Stopwatch t = new Stopwatch();

t.Start();

foreach (string str in list)

{

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

//Если расстояние меньше порогового, то слово добавляется в результат

if (dist <= maxDist)

{

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

}

}

t.Stop();

this.labelSearch.Text= t.Elapsed.ToString();

this.listBoxResult.BeginUpdate();

//Очистка списка

this.listBoxResult.Items.Clear();

//Вывод результатов поиска

foreach (var x in tempList)

{

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

this.listBoxResult.Items.Add(temp);

}

this.listBoxResult.EndUpdate();

}

else

{

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

}

}



Диаграмма классов:

