|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**Факультет «Радиотехнический»**

**Кафедра «Системы обработки информации и управления»**

Лабораторная работа № 5

по дисциплине «Базовые компоненты интернет технологий»

Выполнил:

студент(ка) группы № РТ5-31Б

Пакало Александр Сергеевич

подпись, дата

Проверил:

преподаватель каф. ИУ5  
Гапанюк Юрий Евгеньевич

подпись, дата

2020 г.

Оглавление

[Описание задания. 3](#_Toc59482161)

[Диаграмма классов. 3](#_Toc59482162)

[Текст программы. 4](#_Toc59482163)

[Program.cs 4](#_Toc59482164)

[LevensteinDistance.WagnerFischerAlgorithm.cs 4](#_Toc59482165)

[Анализ результатов. 5](#_Toc59482166)

# Описание задания.

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.



# Текст программы.

## Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab\_5

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

## LevensteinDistance.WagnerFischerAlgorithm.cs

using System;

namespace LevensteinDistance.WagnerFischerAlgorithm

{

static public class DamerauLevenstein

{

/// <summary>

/// Вычисление расстояния Дамерау-Левенштейна.

/// </summary>

static public int CalculateDistance(string s1, string s2)

{

if ((s1 == null) || (s2 == null)) return -1;

int s1Len = s1.Length;

int s2Len = s2.Length;

// Если хотя бы одна строка пустая, возвращается длина другой строки.

if (s1Len == 0) return s2Len;

if (s2Len == 0) return s1Len;

// Приведение строк к верхнему регистру.

string str1 = s1.ToUpper();

string str2 = s2.ToUpper();

// Объявление матрицы.

int[,] matrix = new int[s1Len + 1, s2Len + 1];

// Инициализация нулевой строки и нулевого столбца матрицы.

for (int i = 0; i <= s1Len; i++) matrix[i, 0] = i;

for (int j = 0; j <= s2Len; j++) matrix[0, j] = j;

// Вычисление расстояния Дамерау-Левенштейна.

for (int i = 1; i <= s1Len; i++)

{

for (int j = 1; j <= s2Len; 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

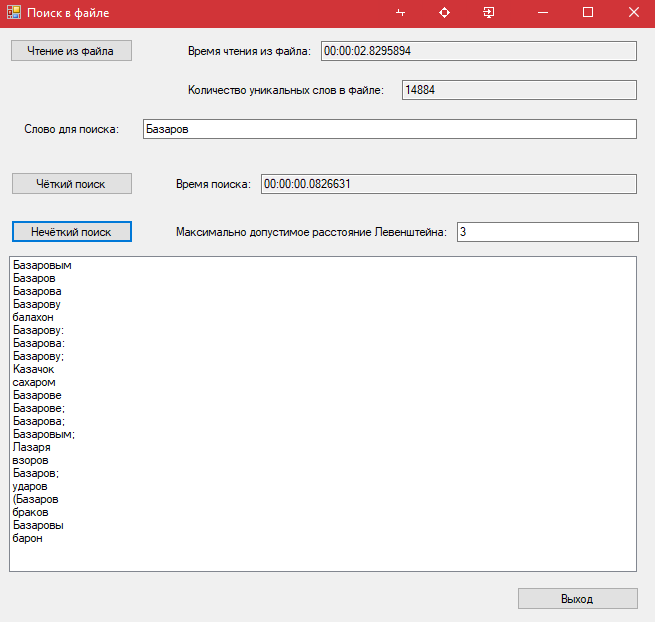
return matrix[s1Len, s2Len];

}

}

}

# Анализ результатов.



Скриншот 1 (демонстрация работы программы нечёткого поиска).