**Отчет по лабораторной работе № 5**

**по курсу «Базовые компоненты интернет технологий»**

Выполнил:

студент группы ИУ5-33

Кондратьев Максим

Дата: 10.12.2017

**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.

**Текст программы на языке C#.**

*Файл Distance.cs*

using System;

namespace DistanceLibrary

{

public static class Distance

{

public static int CalculateDistance(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];

}

}

}

*Файл Form1.cs*

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

using System.Windows.Forms;

using DistanceLibrary;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

private void Button\_fileLoad1(object sender, EventArgs e)

{

OpenFileDialog file = new OpenFileDialog();

if (file.ShowDialog() == DialogResult.OK)

{

Stopwatch time = new Stopwatch();

time.Start();

string text = File.ReadAllText(file.FileName);

textBox1.Text = text;

char[] seps = new char[] { ' ', '.', ',', '!', '?', '/', '\t', '\n', '(', ')' };

string[] textArray = text.Split(seps);

foreach (string word in textArray)

{

string trimmedWord = word.Trim();

if (!words.Contains(trimmedWord))

{

words.Add(trimmedWord);

}

}

time.Stop();

label\_time.Text = "Затраченное время на загрузку файла: " + time.Elapsed.ToString();

}

else

{

MessageBox.Show("Что-то не так!");

}

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button\_findWord\_Click(object sender, EventArgs e)

{

string wordToFind = textBox\_wordToFind.Text.Trim();

textBox\_wordToFind.Text = wordToFind;

int maxDistance = Convert.ToInt32(textBox\_maxDistance.Text);

if (!string.IsNullOrEmpty(wordToFind) && words.Count > 0)

{

string wordToFindUpper = wordToFind.ToUpper();

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

Stopwatch time = new Stopwatch();

time.Start();

foreach (string str in words)

{

if (Distance.CalculateDistance(str.ToUpper(), wordToFindUpper) <= maxDistance)

{

tempList.Add(str);

}

}

time.Stop();

label\_time.Text = "Затраченное время на поиск слова: " + time.Elapsed.ToString();

listBox.BeginUpdate();

listBox.Items.Clear();

foreach (string str in tempList)

{

listBox.Items.Add(str);

}

listBox.EndUpdate();

}

else

{

MessageBox.Show("Что-то не то!");

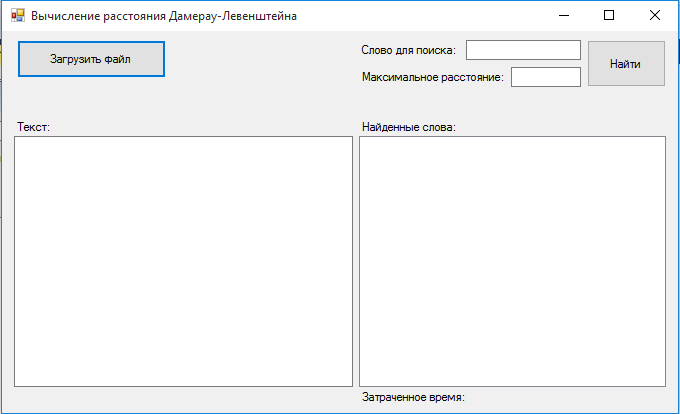
}

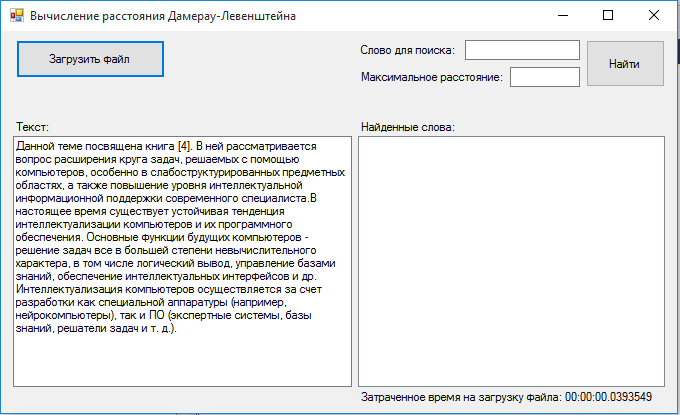
}

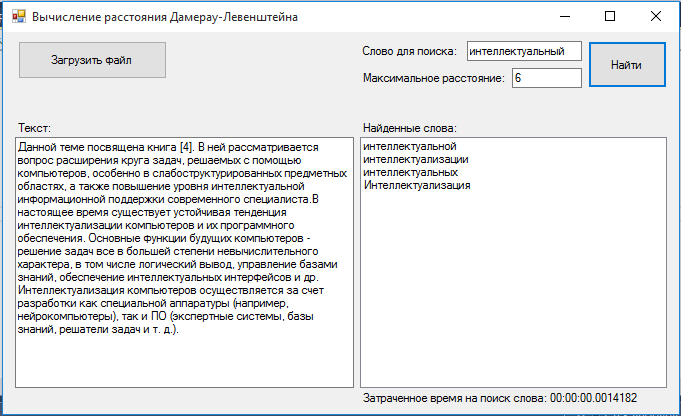
}

}

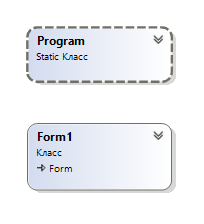
**Результаты выполнения программы:**

****

****

****

**Диаграмма классов:**

****