**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №5

«Трек курса «Задания на основе языка C#»»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-34Б |  | преподаватель каф. ИУ5 |
| Киреев А.А. |  | Ю.Е. Гапанюк. |
| Подпись и дата:  28.12.2020 |  | Подпись и дата:  28.12.2020 |

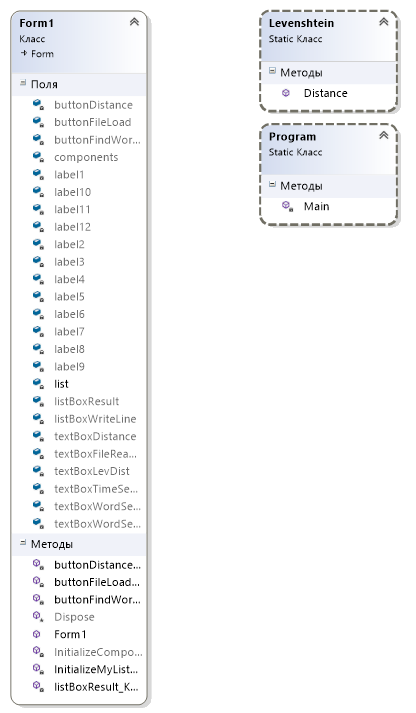
Москва, 2020 г.

**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**

Для Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab5\_1

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

public static class Levenshtein

{

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];

}

}

}

Для Form1.cs

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 Lab5\_1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

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

{

this.listBoxResult.Items.Clear();

OpenFileDialog fileDialog = new OpenFileDialog();//диалог открытия файла

fileDialog.Filter = "текстовые файлы|\*.txt";// фильтрация только по текстовым файлам

fileDialog.InitialDirectory = "C:\\Users\\111\\Documents"; //директория по умолчанию

if (fileDialog.ShowDialog() == DialogResult.OK) //если был выбран файл

{

list.Clear(); //очистка списка от предыдущих файлов

Stopwatch timeOpen = new Stopwatch();

timeOpen.Start();

string text = File.ReadAllText(fileDialog.FileName);//чтение файла

char[] separators = new char[] { ' ', '.', ',', '!', '?', '/', ':', '\t', '\n', '\r' }; //разделители

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

foreach (string strTemp in textArray)

{

string str = strTemp.Trim(separators);//Удаление разделителей в начале и в конце

if (!list.Contains(str)) list.Add(str); //добавление уникальных строк

}

if (list.Contains("") == true) //удаление "слов" при дублировании разделителей

{

list.Remove("");

}

timeOpen.Stop();

this.textBoxFileReadTime.Text = timeOpen.Elapsed.ToString();

foreach (string str in list) //Вывод результатов поиска

{

this.listBoxResult.Items.Add(str);

}

}

else

{

MessageBox.Show("Выберите файл!"); //если файл не выбран

}

}

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

{

char[] separators = new char[] { ' ', '.', ',', '!', '?', '/', ':', '\t', '\n', '\r' }; //разделители

string word1 = this.textBoxWordSearch1.Text.Trim(separators); //Слово1 для поиска

string word2 = this.textBoxWordSearch2.Text.Trim(separators); //Слово2 для поиска

if (!string.IsNullOrWhiteSpace(word1) && !string.IsNullOrWhiteSpace(word1) && list.Count > 0) //Проверка является ли слово пустым или разделителем

{

string wordLower1 = word1.ToLower();

string wordLower2 = word2.ToLower();

int matr = Levenshtein.Distance(wordLower1, wordLower2);

this.textBoxDistance.Text = matr.ToString();

}

}

private void InitializeMyListBox()

{

listBoxResult.SelectionMode = SelectionMode.MultiExtended;

}

private void listBoxResult\_KeyDown\_1(object sender, KeyEventArgs e)

{

if (e.Control == true && e.KeyCode == Keys.C)

{

string s = listBoxResult.SelectedItem.ToString();

Clipboard.SetData(DataFormats.StringFormat, s);

}

}

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

{

this.listBoxWriteLine.Items.Clear();

int maxDist;

Stopwatch timeSearch = new Stopwatch();

timeSearch.Start();

if (!int.TryParse(textBoxLevDist.Text, out maxDist)) {

maxDist = 0;

this.textBoxLevDist.Text = "0";

}

for (int i = 0; i < list.Count(); i++)

{

for (int j = i+1; j < list.Count(); j++)

{

int matr = Levenshtein.Distance(list[i], list[j]);

if (matr < maxDist)

{

string buff = "'" + list[i] + "','" + list[j] + "' -> " + matr.ToString();

this.listBoxWriteLine.Items.Add(buff);

}

}

}

timeSearch.Stop();

this.textBoxTimeSearch.Text = timeSearch.Elapsed.ToString();

}

}

}

**Экранные формы с примерами выполнения программы**

