**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет технологий»

Отчет по лабораторной работе №5

«Вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера»

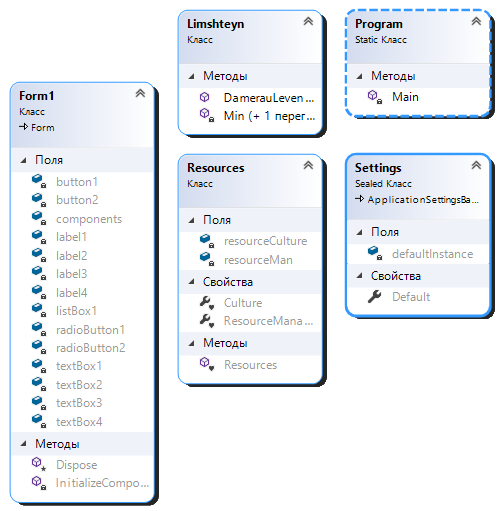
|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б |  | преподаватель каф. ИУ5 |
| Олейников И.И. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

**Условие 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:**

Class1

using System;

namespace \_5

{

public class Limshteyn

{

static int Min(int a, int b, int c)//функция нахождения мин элемента

{

if (a > b)

a = b;

if (a > c)

a = c;

return a;

}

static int Min(int a, int b) => a < b ? a : b;//лямда-вырожение нахождения мин элемента

#region Расстояние Дамерау-Левенштейна

public int DamerauLevenshteinDist(string word1, string word2, bool limht)

{

var n = word1.Length + 1;//размер матрицы по вертикали

var m = word2.Length + 1;//размер матрицы по горизонтали

var mD = new int[n, m];//создаем матрицу

for (var i = 0; i < n; i++)//заполняем первый столбец матрицы по вертикали

mD[i, 0] = i;

for (var j = 0; j < m; j++)//заполяем первый столбец матрицы по горизонтали

mD[0, j] = j;

for (var i = 1; i < n; i++)

{

for (var j = 1; j < m; j++)

{

int costWord = word1[i - 1] == word2[j - 1] ? 0 : 1;//равна ли буква 1 слова букве 2, если да то 0 если нет то 1

mD[i, j] = Min(mD[i - 1, j] + 1 /\*удаление\*/, mD[i, j - 1] + 1 /\* вставка\*/, mD[i - 1, j - 1] + costWord /\* замена\*/);

if (i > 1 && j > 1 && word1[i - 1] == word2[j - 2] && word1[i - 2] == word2[j - 1]&&!limht)//если это не 1 буквы и они "перепутыны крест накрест" и это не метод Лимштейна

mD[i, j] = Min(mD[i, j], mD[i - 2, j - 2] + costWord); // если число меньше посчит до этого, то записываем его

}

}

return mD[n - 1, m - 1];//выводим расстояние

}

#endregion

}

}

Form1

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using System.Diagnostics;

using \_5;

namespace \_4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

bool limsht = true;//Левенштейн или Дамерау-Левенштейн

List<string> list = new List<string>();//создаем лист

private void button1\_Click(object sender, EventArgs e)//открытие файла

{

OpenFileDialog opendial = new OpenFileDialog();//экз. класса

opendial.InitialDirectory = @"";//указываем начальную дерикторию

opendial.Filter = "Файлы txt \*.txt|\*.txt";//ставим фильтир только на txt

opendial.ReadOnlyChecked = true;// режим только для чтения

Stopwatch stopWatch = new Stopwatch();//экз. класса

if (opendial.ShowDialog() == DialogResult.OK)//если диал. окно открыто

{

stopWatch.Start();//запускаем отсчет времени

string str = File.ReadAllText(opendial.FileName);//записываем в переменную стринг текст из файла

string[] s\_mas = str.Split();//записываем в массив стринг слова из str

foreach(string o in s\_mas)

if (!list.Contains(o) && o != "")//если в листе нет такой строки и она не путая

list.Add(o.ToLower());//заносим в лист делая все слова маленькими

stopWatch.Stop();//останавливаем счетчик времени

textBox1.Text = stopWatch.Elapsed.ToString();//вводим в текстбокс сколько по времени это считалось

#region появление кнопок

textBox2.Visible = textBox3.Visible = textBox4.Visible = true;

listBox1.Visible = true;

label2.Visible= label3.Visible= label4.Visible = true;

button2.Visible = true;

radioButton1.Visible = radioButton2.Visible = true;

Width = 700;

#endregion

}

}

private void button2\_Click(object sender, EventArgs e)// поиск элемента

{

Stopwatch stopWatch = new Stopwatch();//экз. класса

stopWatch.Start();//запускаем отсчет времени

string s = textBox3.Text.ToLower();//добавляем сод. текст бокса маленькими буквами

if (!int.TryParse(textBox4.Text, out int dLim)||dLim<0)//пока вводится не число выдаем ошибку, иначе записываем в переменную

dLim = 2;//если проблемы с вводом приравниваем к 2

Limshteyn lim = new Limshteyn();//создаем экз. нашего класса

int t;

foreach (string o in list)//ищем его

if ((t=lim.DamerauLevenshteinDist(o, s, limsht/\*смотря каким способом хотим искать\*/)) <= dLim)//если растояние меньше или равно нужному

{

listBox1.BeginUpdate();//

listBox1.Items.Add($"{s} - {o} - {t}");// добавляем слово

listBox1.EndUpdate();//

stopWatch.Stop();//останавливаем подсчет времени

textBox2.Text = stopWatch.Elapsed.ToString();//выводим время

break;

}

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

listBox1.Items.Clear();//по наж. на листбокс он очищается

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

radioButton2.Checked = !radioButton1.Checked;

limsht = true;//используем метод Лемштейна

label3.Text= "Максимальное\nрасстояние Левенштейна";

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

radioButton1.Checked = !radioButton2.Checked;

limsht = false;//используем метод Дамерау-Лемштейна

label3.Text = "Максимальное расстояние\nДамерау-Левенштейна";

}

}

}

**Примеры выполнения программы:**

