**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

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Ветошкин Артём |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

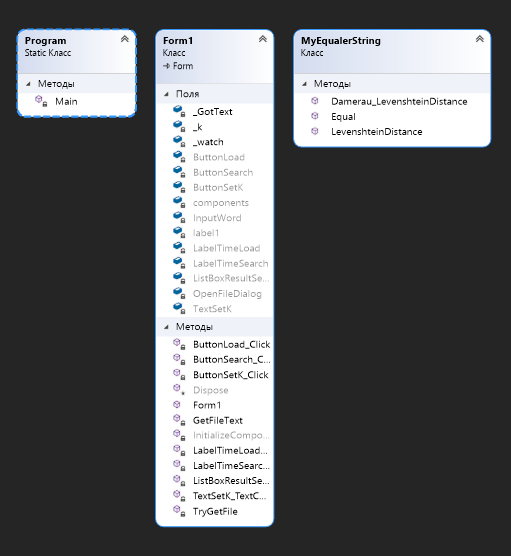
Москва, 2020 г.

**Постановка задачи**

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**

Файл CompareString.cs:

using System;

namespace CompareString {

class MyEqualerString {

static public long LevenshteinDistance(string s1, string s2, long k = long.MaxValue) {

if (s1.Length is 0) return s2.Length;

if (s2.Length is 0) return s1.Length;

int size = s2.Length + 1;

bool use\_optimaze = !k.Equals(long.MaxValue);

long[][] D = new long[2][] { new long[size], new long[size] };

D[0][0] = 0;

D[1][0] = 1;

for (int j = 1; j <= s2.Length; ++j) {

D[0][j] = j;

}

for (int i = 1; i <= s1.Length; ++i) {

for (int j = 1; j <= s2.Length; ++j) {

D[1][j] = Math.Min(D[0][j] + 1, Math.Min(D[1][j - 1] + 1,

D[0][j - 1] + (s1[i - 1] == s2[j - 1] ? 0 : 1)));

}

D[1].CopyTo(D[0], 0);

D[1][0] = i + 1;

if (use\_optimaze) {

if (D[1][Math.Min(i, size - 1)] > k) {

return k;

}

}

}

return D[Math.Min(1, s1.Length - 1)][size - 1];

}

static public long Damerau\_LevenshteinDistance(string s1, string s2, long k = long.MaxValue) {

if (s1.Length is 0) return s2.Length;

if (s2.Length is 0) return s1.Length;

int size = s2.Length + 1;

bool use\_optimaze = !k.Equals(long.MaxValue);

long[][] D = new long[3][] { new long[size],

new long[size], new long[size] };

D[0][0] = 0;

D[1][0] = 1;

D[2][0] = 2;

for (int j = 1; j <= s2.Length; ++j) {

D[0][j] = j;

}

for (int j = 1; j <= s2.Length; ++j) {

D[1][j] = Math.Min(D[0][j] + 1, Math.Min(D[1][j - 1] + 1,

D[0][j - 1] + (s1[0] == s2[j - 1] ? 0 : 1)));

}

for (int i = 2; i <= s1.Length; ++i) {

for (int j = 1; j <= s2.Length; ++j) {

if (j == 0) continue;

D[2][j] = Math.Min(D[1][j] + 1, Math.Min(D[2][j - 1] + 1,

Math.Min(D[1][j - 1] + (s1[i - 1] == s2[j - 1] ? 0 : 1)

, ((j - 2 >= 0 && s1[i - 1] == s2[j - 2]

&& s2[j - 1] == s1[i - 2])

? D[0][j - 2] + 1 : int.MaxValue))));

}

D[1].CopyTo(D[0], 0);

D[2].CopyTo(D[1], 0);

D[2][0] = i + 1;

if (use\_optimaze) {

if (D[2][Math.Min(i, size - 1)] > k) {

return k;

}

}

}

return D[s1.Length < 2 ? 1 : 2][size - 1];

}

static public bool Equal(string s1, string s2,

Func<string, string, long, long> get\_distance,

long k = 0) {

return get\_distance(s1, s2, k + 1) <= k;

}

}

}

Файл Program.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace IT4 {

static class Program {

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main() {

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Файл Form1.cs:

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using System.IO;

using System.Diagnostics;

using CompareString;

namespace IT4 {

public partial class Form1 : Form {

private List<string> \_GotText = new List<string>();

private long \_k = 0;

private Stopwatch \_watch = new Stopwatch();

public Form1() {

InitializeComponent();

TextSetK.Visible = false;

ButtonSetK.Visible = false;

ButtonSearch.Visible = false;

LabelTimeSearch.Visible = false;

ListBoxResultSearch.Visible = false;

LabelTimeLoad.Visible = false;

InputWord.Visible = false;

Height = 120;

}

private bool TryGetFile(string filter) {

var filePath = string.Empty;

OpenFileDialog.InitialDirectory = "c:\\";

OpenFileDialog.Filter = filter;

OpenFileDialog.FilterIndex = 2;

OpenFileDialog.RestoreDirectory = true;

return OpenFileDialog.ShowDialog() == DialogResult.OK;

}

private List<string> GetFileText() {

if(!TryGetFile("txt files (\*.txt)|\*.txt")) {

return null;

}

\_watch.Start();

var fileStream = OpenFileDialog.OpenFile();

var fileContent = string.Empty;

using (StreamReader reader = new StreamReader(fileStream)) {

fileContent = reader.ReadToEnd();

}

var got\_list = fileContent.Split(" ,.;?!\"\'\n\t«»)(".ToCharArray());

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

foreach(string word in got\_list) {

if(!list.Contains(word)) {

list.Add(word);

}

}

return list;

}

private void ButtonLoad\_Click(object sender, EventArgs e) {

\_GotText = GetFileText();

\_watch.Stop();

if(\_GotText is null) {

return;

}

LabelTimeLoad.Text = "Затраченое время: "

+ \_watch.ElapsedTicks.ToString() + "ticks";

if (Height > 170) return;

ButtonSetK.Visible = true;

LabelTimeLoad.Visible = true;

TextSetK.Visible = true;

Height = 170;

}

private void ButtonSetK\_Click(object sender, EventArgs e) {

if(TextSetK.Text.Length is 0) {

return;

}

if (!long.TryParse(TextSetK.Text, out \_k)) {

\_k = 0;

}

if (Height > 200) return;

ButtonSearch.Visible = true;

InputWord.Visible = true;

Height = 200;

}

private void ButtonSearch\_Click(object sender, EventArgs e) {

string word = InputWord.Text;

\_watch.Start();

ListBoxResultSearch.BeginUpdate();

ListBoxResultSearch.Items.Clear();

foreach (var check\_word in \_GotText) {

if (MyEqualerString.Equal(check\_word, word, MyEqualerString.Damerau\_LevenshteinDistance, \_k)) {

ListBoxResultSearch.Items.Add(check\_word);

}

}

ListBoxResultSearch.EndUpdate();

\_watch.Stop();

LabelTimeSearch.Text = "Затраченое время: "

+ \_watch.ElapsedTicks.ToString() + "ticks";

LabelTimeSearch.Visible = true;

ListBoxResultSearch.Visible = true;

Height = 294;

}

//-----------------------------------------------------------------------------------

private void ListBoxResultSearch\_SelectedIndexChanged(object sender, EventArgs e) {

}

private void LabelTimeLoad\_Click(object sender, EventArgs e) {

}

private void LabelTimeSearch\_Click(object sender, EventArgs e) {

}

private void TextSetK\_TextChanged(object sender, EventArgs e) {

}

}

}

Файл с текстом:

Текст (от лат. textus — ткань; сплетение, сочетание) — зафиксированная на каком-либо материальном носителе человеческая мысль; в общем плане связная и полная последовательность символов.

Существуют две основные трактовки понятия «текст»: имманентная (расширенная, философски нагруженная) и репрезентативная (более частная). Имманентный подход подразумевает отношение к тексту как к автономной реальности, нацеленность на выявление его внутренней структуры. Репрезентативный — рассмотрение текста как особой формы представления знаний о внешней тексту действительности.

В лингвистике термин «текст» используется в широком значении, включая и образцы устной речи. Восприятие текста изучается в рамках лингвистики текста и психолингвистики. Так, например, И. Р. Гальперин определяет текст следующим образом: «Это письменное сообщение, объективированное в виде письменного документа, состоящее из ряда высказываний, объединённых разными типами лексической, грамматической и логической связи, имеющее определённый моральный характер, прагматическую установку и соответственно литературно обработанное»[1].

**Анализ результатов**

