**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 |
| Корчевский Александр |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

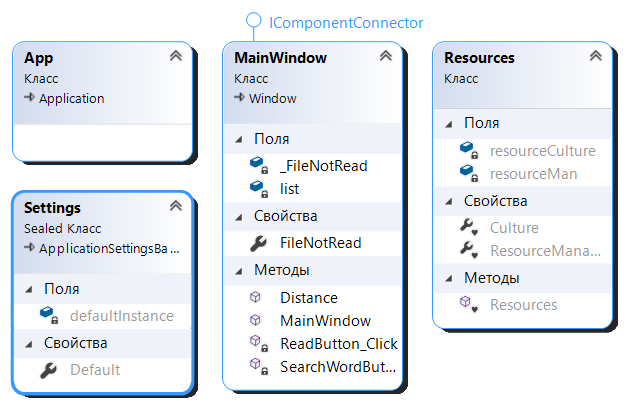
Москва, 2020 г.

**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:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System;

using System.IO;

using System.Windows;

using Microsoft.Win32;

using System.Diagnostics;

using System.Threading;

namespace lab05

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public static int Distance(string str1P, string str2P)

{

if ((str1P == null) || (str2P == null)) return -1;

int len1 = str1P.Length;

int len2 = str2P.Length;

if ((len1 == 0) && (len2 == 0)) return 0;

if (len1 == 0) return len2;

if (len2 == 0) return len1;

string str1 = str1P.ToUpper();

string str2 = str2P.ToUpper();

int[,] matrix = new int[len1 + 1, len2 + 1];

for (int i = 0; i <= len1; i++) matrix[i, 0] = i;

for (int i = 0; i <= len2; i++) matrix[0, i] = i;

for (int i = 1; i <= len1; i++)

{

for (int j = 1; j <= len2; j++)

{

int eq = (

(str1.Substring(i - 1, 1) ==

str2.Substring(j - 1, 1)) ? 0 : 1);

int ins = matrix[i, j - 1] + 1;

int del = matrix[i - 1, j] + 1;

int subst = matrix[i - 1, j - 1] + eq;

matrix[i, j] = Math.Min(Math.Min(ins, del), subst);

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] + eq);

}

}

}

return matrix[len1, len2];

}

public MainWindow()

{

InitializeComponent();

}

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

private bool \_FileNotRead = true;

public bool FileNotRead

{

get => \_FileNotRead;

set

{

\_FileNotRead = value;

}

}

private void ReadButton\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog fd = new OpenFileDialog();

fd.Filter = "Text files (\*.txt)|\*.txt";

if (fd.ShowDialog() == true)

{

Stopwatch timer = new Stopwatch();

timer.Start();

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

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

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

foreach (string strItem in textArray)

{

string str = strItem.Trim();

if (list.Contains(str) == false)

{

list.Add(str);

}

}

timer.Stop();

ReadTimeTextBox.Text = timer.Elapsed.ToString() + list.Count.ToString();

FileNotRead = false;

}

else

{

MessageBox.Show("Выберите файл!");

}

}

private void SearchWordButton\_Click(object sender, RoutedEventArgs e)

{

if (FileNotRead)

{

MessageBox.Show("Выберите файл!");

return;

}

string word = SearchWordTextBox.Text.Trim();

if (!string.IsNullOrWhiteSpace(word) && list.Count > 0)

{

Int32.TryParse(MaxDistanceTextBox.Text, out int max);

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

Stopwatch timer = new Stopwatch();

timer.Start();

foreach (string str in list)

{

int dist = Distance(str, word);

if (dist <= max)

{

if ((bool)YesRadioButton.IsChecked)

{

tempList.Add(str + " - " + dist.ToString());

}

else

{

tempList.Add(str);

}

}

}

timer.Stop();

SearchTimeTextBox.Text = timer.Elapsed.ToString();

WordList.Items.Clear();

foreach (string str in tempList)

{

WordList.Items.Add(str);

}

}

else

{

MessageBox.Show("Выберите файл и введите слово для поиска");

}

}

}

}

<Window x:Class="lab05.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:lab05"

xmlns:fa5="http://schemas.fontawesome.com/icons/"

mc:Ignorable="d"

Title="Поиск в файле" Height="500" Width="470">

<Grid Background="LightCoral">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="170"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="40"/>

<RowDefinition Height="40"/>

<RowDefinition Height="40"/>

<RowDefinition Height="40"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<Button Width="100" Click="ReadButton\_Click" Background="MediumAquamarine" Height="20" Margin="10" VerticalAlignment="Center" HorizontalAlignment="Left">Выбор файла</Button>

<StackPanel VerticalAlignment="Center" Orientation="Horizontal" Grid.Column="1">

<Label VerticalAlignment="Center" Margin="5,0,10,0">Время чтения из файла:</Label>

<TextBox Name="ReadTimeTextBox" IsReadOnly="True" Width="110" Height="20"></TextBox>

</StackPanel>

<Label Grid.Row="1" VerticalAlignment="Center" HorizontalAlignment="Left" Margin="5">Слово для поиска:</Label>

<StackPanel Orientation="Horizontal" VerticalAlignment="Center" Grid.Row="1" Grid.Column="1">

<TextBox Name="SearchWordTextBox" HorizontalAlignment="Left" Width="110" Height="20" VerticalAlignment="Center" Margin="10"></TextBox>

<Button Name="SearchButton" Click="SearchWordButton\_Click" Background="MediumAquamarine" Height="20" Width="60" VerticalAlignment="Center">

<fa5:ImageAwesome Height="15" Icon="Solid\_Search"/>

</Button>

</StackPanel>

<Label Grid.Row="2" VerticalAlignment="Center" HorizontalAlignment="Left" Margin="5">Максимальное расстояние:</Label>

<TextBox Name="MaxDistanceTextBox" Width="110" Height="20" Grid.Row="2" Grid.Column="1" HorizontalAlignment="Left" Margin="10"></TextBox>

<Label Grid.Row="3" VerticalAlignment="Center" HorizontalAlignment="Left" Margin="5">Время поиска:</Label>

<TextBox Name="SearchTimeTextBox" IsReadOnly="True" Width="110" Height="20" Grid.Row="3" Grid.Column="1" HorizontalAlignment="Left" Margin="10"></TextBox>

<StackPanel HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" Grid.Row="2" Grid.Column="1" Grid.RowSpan="2">

<Label FontSize="12">Показывать расстояние?</Label>

<RadioButton x:Name="YesRadioButton" IsChecked="True" Margin="5">Да</RadioButton>

<RadioButton Margin="5">Нет</RadioButton>

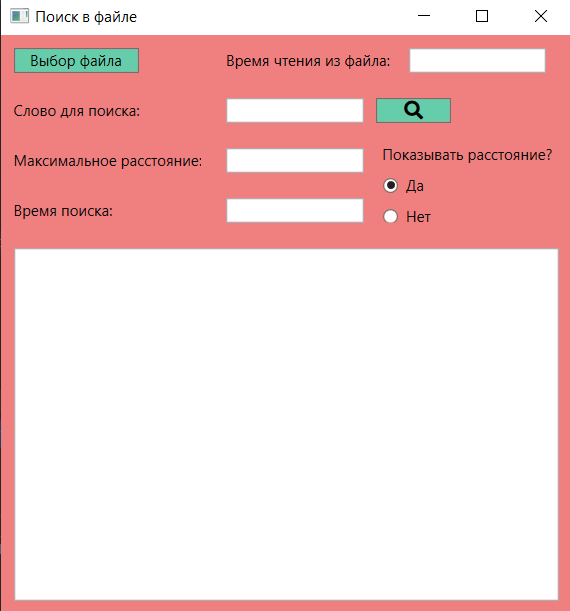
</StackPanel>

<ListBox Name="WordList" Grid.ColumnSpan="2" Grid.Row="4" Margin="10"></ListBox>

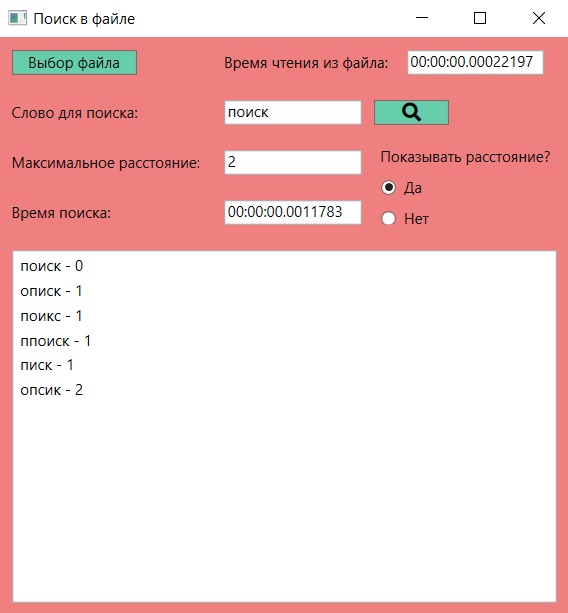
</Grid>

</Window>

**Результаты:**



Расстояние Дамерау-Левенштейна вычисляется с помощью алгоритма Вагнера-Фишера, его можно задать самостоятельно.



Также можно настроить так, чтобы расстояние не выводилось.

