**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №4

«Работа с файлами»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-31 |  | преподаватель каф. ИУ5 |
| Агличеев Михаил |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

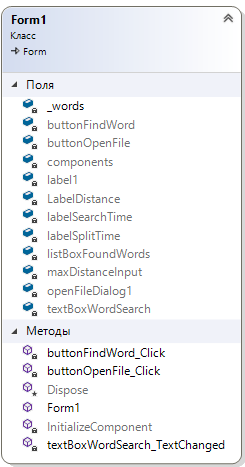
Москва, 2020г.

1. Описание задания

Разработать программу, реализующую работу с файлами.

1. Программа должна быть разработана в виде приложения Windows Forms на языке C#. По желанию вместо Windows Forms возможно использование WPF.
2. Добавить кнопку, реализующую функцию чтения файла в список слов List<string>.
3. Для выбора имени файла используется класс OpenFileDialog, который открывает диалоговое окно с выбором файла. Ограничить выбор только файлами с расширением «.txt».
4. Для чтения из файла рекомендуется использовать статический метод ReadAllText() класса File (пространство имен System.IO). Содержимое файла считывается методом ReadAllText() в виде одной строки, далее делится на слова с использованием метода Split() класса string. Слова сохраняются в список List<string>.
5. При сохранении слов в список List<string> дубликаты слов не записываются. Для проверки наличия слова в списке используется метод Contains().
6. Вычислить время загрузки и сохранения в список с использованием класса Stopwatch (пространство имен System.Diagnostics). Вычисленное время вывести на форму в поле ввода (TextBox) или надпись (Label).
7. Добавить на форму поле ввода для поиска слова и кнопку поиска. При нажатии на кнопку поиска осуществлять поиск введенного слова в списке. Слово считается найденным, если оно входит в элемент списка как подстрока (метод Contains() класса string).
8. Добавить на форму список (ListBox). Найденные слова выводить в список с использованием метода «название\_списка.Items.Add()». Вызовы метода «название\_списка.Items.Add()» должны находится между вызовами методов «название\_списка.BeginUpdate()» и «название\_списка. EndUpdate()».
9. Вычислить время поиска с использованием класса Stopwatch. Вычисленное время вывести на форму в поле ввода (TextBox) или надпись (Label).

2. UML-диаграмма классов



3. Листинг программы

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace lab4

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Form.cs

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 static Лабораторная\_\_\_5.Distance;

namespace lab4

{

public partial class Form : System.Windows.Forms.Form

{

public Form()

{

InitializeComponent();

\_words = new List<string>();

}

private List<string> \_words;

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

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

labelSearchTime.Text = "Time taken to find word: ";

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

string fileText = File.ReadAllText(openFileDialog1.FileName);

\_words.Clear();

foreach (string word in fileText.Split(' '))

{

if (!\_words.Contains(word))

{

\_words.Add(word);

}

}

stopwatch.Stop();

labelSplitTime.Text = String.Format("Time taken to split file: {0:0.00} s", stopwatch.Elapsed.Seconds);

}

}

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

{

labelSearchTime.Text = "Word not found!";

LabelDistance.Text = "Levenstain distance:";

string wordToFind = textBoxWordSearch.Text;

if (!string.IsNullOrEmpty(wordToFind))

{

if (!listBoxFoundWords.Items.Contains(wordToFind))

{

int maxDistance = (int) maxDistanceInput.Value;

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

foreach (string word in \_words)

{

int currentDistance = Levenshtain(word, wordToFind);

if (currentDistance <= maxDistance) // Checking if Levenstain distance is too big

{

listBoxFoundWords.BeginUpdate();

listBoxFoundWords.Items.Add(wordToFind);

listBoxFoundWords.EndUpdate();

stopwatch.Stop();

labelSearchTime.Text = String.Format("Time taken to find word: {0:0.00} ms", stopwatch.Elapsed.Milliseconds);

LabelDistance.Text = String.Format("Levenstain distance: {0}", currentDistance);

break;

}

}

}

else

{

labelSearchTime.Text = "Word has already been found!";

}

}

}

private void textBoxWordSearch\_TextChanged(object sender, EventArgs e)

{

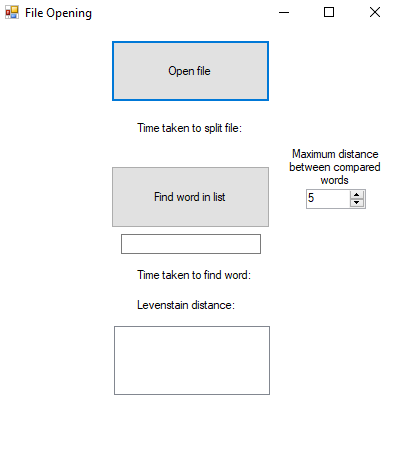
}

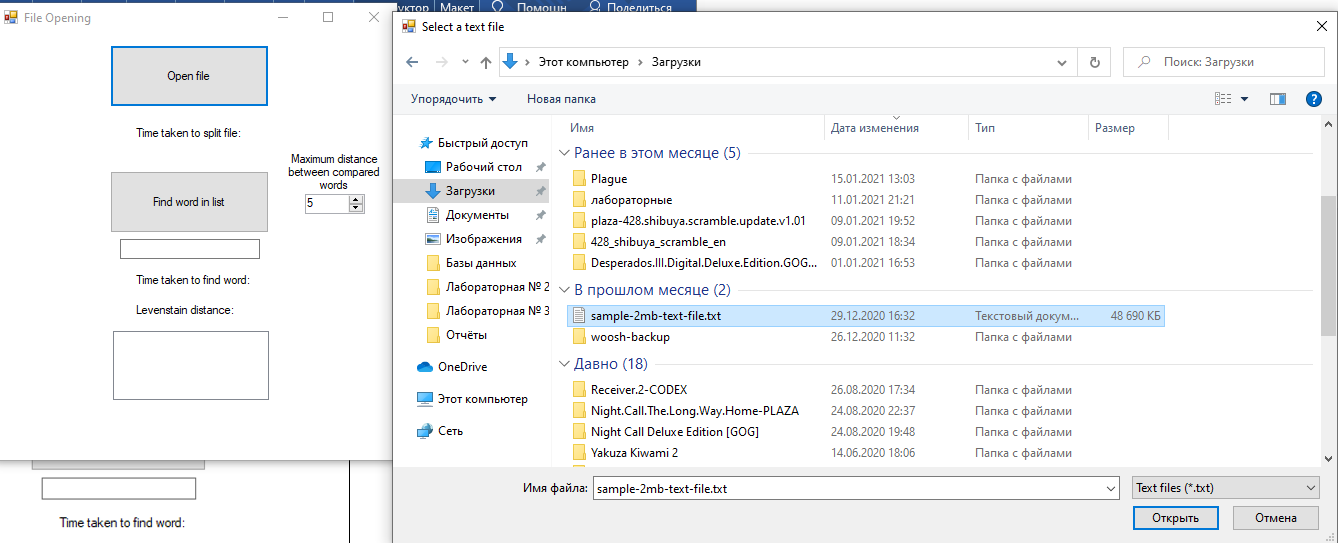
}

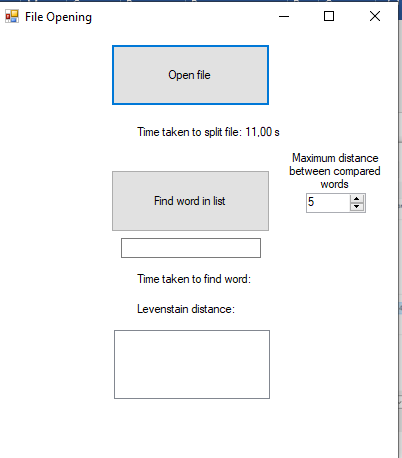
}

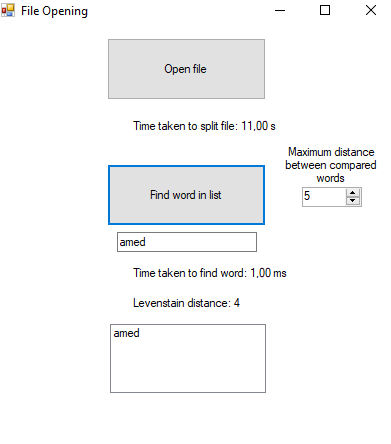
Вычисление расстояние Левенштейна (содержимое Distance.dll) – см. отчёт по лабораторной № 5

4. Результаты

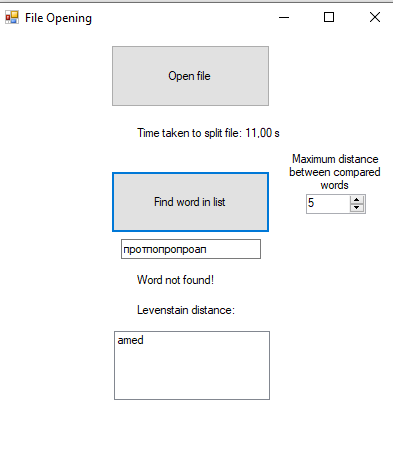


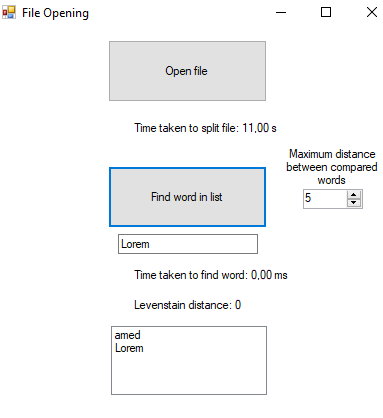






(На деле, такого слова нет в исходном тексте, но программа гарантированно «находит» его, если длина искомого слова меньше заданного расстояния Левенштейна)





Слово “Lorem” в тексте присутствует, причём оно там первое, поэтому расстояние Левенштейна нулевое.