**Отчёт по лабораторной работе №5 по предмету «Базовые компоненты интернет-технологий»**

Подготовила: Гладова Анастасия

Группа: ИУ5-31

МГТУ имени Н.Э.Баумана

2017 г.

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).

**Диаграмма классов:**

**Текст программы:**

**Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Diagnostics;

using EditDistanceProject;

namespace bkit\_lab\_5

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

label1.Visible = false;

textBoxFind.Visible = false;

buttonExact.Visible = false;

label2.Visible = false;

listBoxResult.Visible = false;

label3.Visible = false;

label4.Visible = false;

//добавилось к элементам 4 лабы

label5.Visible = false;

label6.Visible = false;

textBoxMaxDist.Visible = false;

buttonApprox.Visible = false;

}

/// <summary>

/// Список слов

/// </summary>

// Для хранения списка слов, прочитанных из файла, в классе Form1 используется поле list:

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

//загрузить файл - код обработчика кнопки

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

{

OpenFileDialog fd = new OpenFileDialog();

fd.Filter = "Текстовые файлы|\*.txt";

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

{

Stopwatch t = new Stopwatch();

t.Start();

//Чтение файла в виде строки

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

//Разделительные символы для чтения из файла

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

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

foreach (string strTemp in textArray)

{

//Удаление пробелов в начале и конце строки

string str = strTemp.Trim();

//Добавление строки в список, если строка не содержится в списке

if (!list.Contains(str)) list.Add(str);

}

t.Stop();

//this.textBoxFileReadTime.Text = t.Elapsed.ToString();

//this.textBoxFileReadCount.Text = list.Count.ToString();

label1.Visible = true;

label1.Text = "FileReadTime: " + t.Elapsed.ToString();

textBoxFind.Visible = true;

buttonExact.Visible = true;

label3.Visible = true;

label5.Visible = true;

textBoxMaxDist.Visible = true;

buttonApprox.Visible = true;

label2.Visible = true;

listBoxResult.Visible = true;

label4.Visible = true;

//this.label2.Text = t.Elapsed.ToString();

//this.listBoxResult.BeginUpdate();

label2.Text = t.Elapsed.ToString();

listBoxResult.BeginUpdate();

}

else MessageBox.Show("Необходимо выбрать файл");

}

//Кнопка «Закрыть окно»:

//Эта кнопка вызывает метод Close для текущего объекта, что приводит к закрытию окна.

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

{

this.Close();

//Application.Exit(); //выход из приложения

}

//поиск слова - код обработчика кнопки

private void buttonExact\_Click\_1(object sender, EventArgs e)

{

//Слово для поиска

string word = textBoxFind.Text.Trim();

//Если слово для поиска не пусто

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

{

//Слово для поиска в верхнем регистре

string wordUpper = word.ToUpper();

//Временные результаты поиска

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

Stopwatch t = new Stopwatch();

t.Start();

foreach (string str in list)

{

if (str.ToUpper().Contains(wordUpper))

{

tempList.Add(str);

}

}

t.Stop();

//Очистка списка

listBoxResult.Items.Clear();

//Вывод результатов поиска 1 вар

//int i = 0;

//foreach (string str in tempList)

//{

// i++;

// listBoxResult.Items.Add(i + ". " + str);

//}

//Вывод результатов поиска 2 вар

foreach (string str in tempList)

{

this.listBoxResult.Items.Add(str);

}

//this.listBoxResult.EndUpdate();

listBoxResult.EndUpdate();

}

else MessageBox.Show("Необходимо выбрать файл и ввести слово для поиска");

}

//поиск с применением расстояния левенштейна - код обработчика кнопки

private void buttonApprox\_Click\_1(object sender, EventArgs e)

{

string word = textBoxFind.Text.Trim();

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

{

int maxDist;

if (!int.TryParse(textBoxMaxDist.Text.Trim(), out maxDist))

{

MessageBox.Show("Необходимо указать максимальное расстояние");

return;

}

if (maxDist < 1 || maxDist > 5)

{

MessageBox.Show("Максимальное расстояние должно быть в диапазоне от 1 до 5");

return;

}

string wordUpper = word.ToUpper();

List<Tuple<string, int>> tempList = new List<Tuple<string, int>>();

Stopwatch t = new Stopwatch();

t.Start();

foreach (string str in list)

{

//Вычисление расстояния Дамерау-Левенштейна

int dist = EditDistance.Distance(str.ToUpper(), wordUpper);

//Если расстояние меньше порогового, то слово добавляется в результат

if (dist <= maxDist)

{

tempList.Add(new Tuple<string, int>(str, dist));

}

}

t.Stop();

label6.Visible = true;

label6.Text = "MethodWordSearchTime: " + t.Elapsed.ToString() + "; Done";

listBoxResult.Visible = true;

listBoxResult.BeginUpdate();

//Очистка списка

listBoxResult.Items.Clear();

//Вывод результатов поиска

int i = 0;

foreach (var x in tempList)

{

i++;

string temp = i + ". " + x.Item1 + " (расстояние=" + x.Item2.ToString() + ")";

listBoxResult.Items.Add(temp);

}

listBoxResult.EndUpdate();

}

else

{

MessageBox.Show("Необходимо выбрать файл и ввести слово для поиска");

}

}

}

}

namespace EditDistanceProject

{

public static class EditDistance

{

/// <summary>

/// Вычисление расстояния Дамерау-Левенштейна

/// </summary>

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

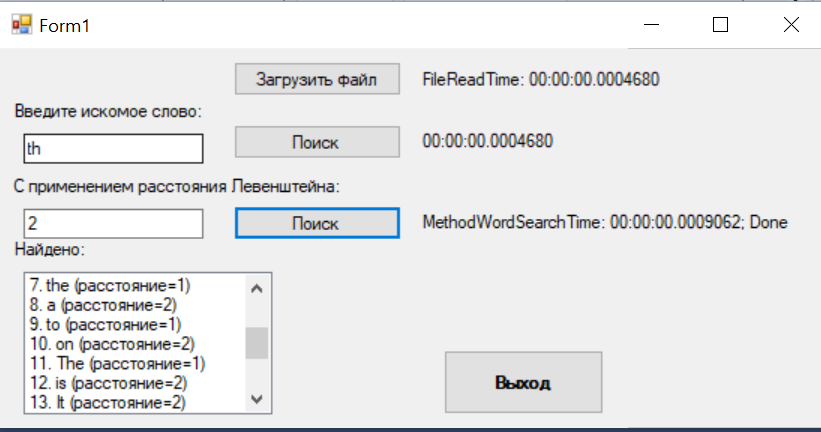
return matrix[str1Len, str2Len];

}

}

}

**Экранные формы с примерами выполнения программы:**

****