|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**Отчёт**

***Лабораторная работа №4 и 5***

Выполнил студент: \_\_***Сафиуллин Антон Павлович\_\_\_\_\_\_\_\_\_\_***

*фамилия, имя, отчество*

Группа: \_\_\_\_***РТ5-31Б\_\_\_\_\_\_\_\_***

**Описание задания**

**Лабораторная работа №4**

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).

**Лабораторная работа №5**

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.

**Текст Программы**

**DamerauLevenshtein.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LabRabota\_4\_BKIT

{

class DamerauLevenshtein

{

static int Minimum(int a, int b) => a < b ? a : b;

static int Minimum(int a, int b, int c) => (a = a < b ? a : b) < c ? a : c;

public static int DamerauLevenshteinDistance(string firstText, string secondText)

{

var n = firstText.Length + 1;

var m = secondText.Length + 1;

var arrayD = new int[n, m];

for (var i = 0; i < n; i++)

{

arrayD[i, 0] = i;

}

for (var j = 0; j < m; j++)

{

arrayD[0, j] = j;

}

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

{

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

{

var cost = firstText[i - 1] == secondText[j - 1] ? 0 : 1;

arrayD[i, j] = Minimum(arrayD[i - 1, j] + 1, // удаление

arrayD[i, j - 1] + 1, // вставка

arrayD[i - 1, j - 1] + cost); // замена

if (i > 1 && j > 1

&& firstText[i - 1] == secondText[j - 2]

&& firstText[i - 2] == secondText[j - 1])

{

arrayD[i, j] = Minimum(arrayD[i, j],

arrayD[i - 2, j - 2] + cost); // перестановка

}

}

}

return arrayD[n - 1, m - 1];

}

}

}

**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.Threading.Tasks;

using System.Windows.Forms;

using System.Diagnostics;

using System.Threading;

using System.IO;

namespace LabRabota\_4\_BKIT

{

public partial class Form1 : Form

{

/// <summary>

/// Список слов

/// </summary>

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

public static string word;

public Form1()

{

InitializeComponent();

textBoxFileReadTime.ReadOnly = true;// Разрешение строк только на чтение

textBoxFileReadCount.ReadOnly = true;

textBoxExactTime.ReadOnly = true;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button7\_Click(object sender, EventArgs e) // выход

{

this.Close();

}

private void button6\_Click(object sender, EventArgs e)// отчёт

{

//Имя файла отчета

string TempReportFileName = "Report\_" +

DateTime.Now.ToString("dd\_MM\_yyyy\_hhmmss");

//Диалог сохранения файла отчета

SaveFileDialog fd = new SaveFileDialog();

fd.FileName = TempReportFileName;

fd.DefaultExt = ".html";

fd.Filter = "HTML Reports|\*.html";

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

{

string ReportFileName = fd.FileName;

//Формирование отчета

StringBuilder b = new StringBuilder();

b.AppendLine("<html>");

b.AppendLine("<head>");

b.AppendLine("<meta http-equiv='Content-Type' content='text/html; charset = UTF - 8'/>");

b.AppendLine("<title>" + "Отчет: " + ReportFileName + "</title>");

b.AppendLine("</head>");

b.AppendLine("<body>");

b.AppendLine("<h1>" + "Отчет: " + ReportFileName + "</h1>");

b.AppendLine("<table border='1'>");

b.AppendLine("<tr>");

b.AppendLine("<td>Время чтения из файла</td>");

b.AppendLine("<td>" + this.textBoxFileReadTime.Text + "</td>");

b.AppendLine("</tr>");

b.AppendLine("<tr>");

b.AppendLine("<td>Количество уникальных слов в файле</td>");

b.AppendLine("<td>" + this.textBoxFileReadCount.Text + "</td>");

b.AppendLine("</tr>");

b.AppendLine("<tr>");

b.AppendLine("<td>Слово для поиска</td>");

b.AppendLine("<td>" + this.textBoxFind.Text + "</td>");

b.AppendLine("</tr>");

b.AppendLine("<tr>");

b.AppendLine("<td>Время поиска</td>");

b.AppendLine("<td>" + this.textBoxExactTime.Text + "</td>");

b.AppendLine("</tr>");

b.AppendLine("<tr>");

b.AppendLine("<tr>");

b.AppendLine("<td>Максимальное расстояние для нечеткого поиска</td>");

b.AppendLine("<td>" + this.textBox1.Text + "</td>");

b.AppendLine("</tr>");

b.AppendLine("<tr>");

b.AppendLine("<td>Время нечеткого поиска</td>");

b.AppendLine("<td>" + this.textBox3.Text + "</td>");

b.AppendLine("</tr>");

b.AppendLine("<tr valign='top'>");

b.AppendLine("<td>Результаты поиска</td>");

b.AppendLine("<td>");

b.AppendLine("<ul>");

foreach (var x in this.listBoxResult.Items)

{

b.AppendLine("<li>" + x.ToString() + "</li>");

}

b.AppendLine("</ul>");

b.AppendLine("</td>");

b.AppendLine("</tr>");

b.AppendLine("</table>");

b.AppendLine("</body>");

b.AppendLine("</html>");

//Сохранение файла

File.AppendAllText(ReportFileName, b.ToString());

MessageBox.Show("Отчет сформирован. Файл: " + ReportFileName);

}

}

private void button3\_Click(object sender, EventArgs e) // открытие файла и чтение

{

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();

}

else

{

MessageBox.Show("Необходимо выбрать файл");

}

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void label7\_Click(object sender, EventArgs e)

{

}

private void label5\_Click(object sender, EventArgs e)

{

}

private void textBox5\_TextChanged(object sender, EventArgs e)

{

}

private void button4\_Click(object sender, EventArgs e) // конпка поиска

{

//Слово для поиска

word = this.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();

this.textBoxExactTime.Text = t.Elapsed.ToString();

this.listBoxResult.BeginUpdate();

//Очистка списка

this.listBoxResult.Items.Clear();

//Вывод результатов поиска

if (tempList.Count==0)

{

MessageBox.Show("Данного слова нет в файле");

}

foreach (string str in tempList)

{

this.listBoxResult.Items.Add(str);

}

this.listBoxResult.EndUpdate();

}

else

{

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

}

}

private void button5\_Click(object sender, EventArgs e)

{

}

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

{

}

private void Form1\_Load\_1(object sender, EventArgs e)

{

}

private void button3\_Click\_1(object sender, EventArgs e)

{

}

private void label4\_Click(object sender, EventArgs e)

{

}

public static void textBoxFind\_TextChanged(object sender, EventArgs e)

{

}

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

{

}

public void listBoxResult\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void button3\_Click\_2(object sender, EventArgs e) // кнопка нечёткий поиск

{

//Слово для поиска

word = this.textBoxFind.Text.Trim();

int range=2;

string s\_range = this.textBox1.Text.Trim();

if (string.IsNullOrEmpty(s\_range))

{

MessageBox.Show("Расстояние установлено на: " + range);

}

else if( !string.IsNullOrEmpty(s\_range) )

{

range = Convert.ToInt32(s\_range);

}

//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 (DamerauLevenshtein.DamerauLevenshteinDistance(word,str)<= range)

{

tempList.Add(str);

}

}

t.Stop();

this.textBox3.Text = t.Elapsed.ToString();

this.listBoxResult.BeginUpdate();

//Очистка списка

this.listBoxResult.Items.Clear();

//Вывод результатов поиска

if (tempList.Count == 0)

{

MessageBox.Show("Данного слова нет в файле");

}

foreach (string str in tempList)

{

this.listBoxResult.Items.Add(str);

}

this.listBoxResult.EndUpdate();

}

else

{

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

}

}

private void textBox1\_TextChanged\_1(object sender, EventArgs e)

{

}

private void textBoxFind\_TextChanged\_1(object sender, EventArgs e)

{

}

}

}

**Экранные формы с примерами выполнения программы**









 