**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)**

**Факультет «Радиотехнический»**

**Кафедра «Системы обработки информации и управления»**

Отчет по лабораторной работе №4-5

по курсу БКИТ

Выполнила:

Студентка Группы РТ5-31Б   
Веденеева Екатерина Ильинична

подпись, дата

Проверил:

подпись, дата

2020г.

1. **Задание**

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).
10. **Код**

Файл Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab4Sem3

{

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.IO; // Для работы с файлами

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;

namespace Lab4Sem3

{

public partial class Form1 : Form

{

/// <summary>

/// Выполняется в параллельном потоке для поиска строк

/// </summary>

public static List<PSearchRes> ArrayThreadTask(object paramObj)

{

PSTP param = (PSTP)paramObj;

//Слово для поиска в верхнем регистре

string wordUpper = param.wordPattern.Trim().ToUpper();

//Результаты поиска в одном потоке

List<PSearchRes> Result = new List<PSearchRes>();

//Перебор всех слов во временном списке данного потока

foreach (string str in param.tempList)

{

//Вычисление расстояния Дамерау-Левенштейна

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

//Если расстояние меньше порогового, то слово добавляется врезультат

if (dist <= param.maxDist)

{

PSearchRes temp = new PSearchRes()

{

word = str,

dist = dist,

ThreadNum = param.ThreadNum

};

Result.Add(temp);

}

}

return Result;

}

public Form1()

{

InitializeComponent();

}

/// <summary>

/// Список слов

/// </summary>

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

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();

}

else

{

MessageBox.Show("Необходимо выбрать файл");

}

}

private void button1\_Click(object sender, EventArgs e)

{

// Слово для поиска

string searchWord = this.textBoxFind.Text.Trim();

this.listBoxResult.Items.Add(searchWord);

this.listBoxResult.Items.Add(list.Count);

//Если поисковая строка не пустая

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

{

int maxL;

// Проверка на дуррака

if (!int.TryParse(this.textBoxMaxL.Text.Trim(), out maxL))

{

MessageBox.Show("Не указано максимальное расстояние. Заполните поле.");

return;

}

// Проверка на дурака (2)

if (maxL < 1 || maxL > 5)

{

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

return;

}

//Проверка на слепого дурака (заполнение количества потоков)

int sQuant= Convert.ToInt32(this.textBoxStreams.Text.Trim());

// Таймер. Старт.

Stopwatch timer = new Stopwatch();

timer.Start();

/////////////////////////// НАЧАЛО ПОИСКА //////////////////////////

//Список для хранения результатов.

List<PSearchRes> Ress = new List<PSearchRes>();

// Разбиваем список на фрагменты для хапуска в разных потоках

List<MinMax> arrDividedList = SubArrays.DivideSubArrays(0, list.Count, sQuant);

int count = arrDividedList.Count;

//Количество потоков соответствует количеству фрагментов массива

Task<List<PSearchRes>>[] tasks = new Task<List<PSearchRes>>[count];

//Запуск потоков

for (int i = 0; i < count; i++)

{

//Создание временного списка, чтобы потоки не работали параллельно с одной коллекцией

List<string> tempTaskList = list.GetRange(arrDividedList[i].Min,

arrDividedList[i].Max - arrDividedList[i].Min);

tasks[i] = new Task<List<PSearchRes>>(

//Метод, который будет выполняться в потоке

ArrayThreadTask,

//Параметры потока

new PSTP()

{

tempList = tempTaskList,

maxDist = maxL,

ThreadNum = i,

wordPattern = searchWord

});

//Запуск потока

tasks[i].Start();

}

Task.WaitAll(tasks);

timer.Stop();

//Объединение результатов

for (int i = 0; i < count; i++)

{

Ress.AddRange(tasks[i].Result);

}

///////////////////////// ЗАВЕРШЕНИЕ ПОИИСКА /////////////////////////

timer.Stop();

///////////////////// ВЫВОД РЕЗУЛЬТАТОВ /////////////////////////////

// Врепя поиска совпадений

this.textBoxSTimer.Text = timer.Elapsed.ToString();

// Вычисленное кол-во потоков

this.textBoxSQuant.Text = sQuant.ToString();

//Очистка списка

this.listBoxResult.Items.Clear();

// Вывод результатов поиска в форму

foreach (var x in Ress)

{

string tmp = x.word + "(расстояние=" + x.dist.ToString() + " поток=" + x.ThreadNum.ToString() + ")";

this.listBoxResult.Items.Add(tmp);

}

// Окончание обновления списка результатов

this.listBoxResult.EndUpdate();

}

else

{

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

}

}

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

{

}

}

}

**3. Результат выполнения программы**

