#### Московский государственный технический университет имени Н.Э. Баумана

## Домашнее задание по дисциплине БКИТ

Выполнил: Байбарин Р.Г., ИУ5-34, 27.12.2017

### 1. Описание задания

Разработать программу, реализующую многопоточный поиск в файле.

1. Программа должна быть разработана в виде приложения Windows Forms на языке C#. По желанию вместо Windows Forms возможно использование WPF.
2. В качестве основы используется макет, разработанный в лабораторных работах №4 и №5.
3. Реализуйте функцию поиска с использованием расстояния Левенштейна в многопоточном варианте. Количество потоков для запуска функции поиска вводится на форме в поле ввода (TextBox). В качестве примера используйте проект «Parallel» из примера «Введение в C#».
4. Реализуйте функцию записи результатов поиска в файл отчета. Файл отчета создается в формате .txt или .html. В качестве примера используйте проект «WindowsFormsFiles» (обработчик события кнопки «Сохранение отчета») из примера «Введение в C#».

### 2. Код программы

using System;

using System.IO;

using System.Diagnostics;

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;

namespace Lab\_5

{

public partial class Form1 : Form

{

public class ParallelSearchResult

{

public string word { get; set; }// Найденное слово

public int dist { get; set; }// Расстояние

public int ThreadNum { get; set; }// Номер потока

public string toString()

{

string result = word + "\_" + dist.ToString() + "\_" + ThreadNum.ToString();

return result;

}

}

public class MinMax

{

public int Min { get; set; }

public int Max { get; set; }

public MinMax(int pmin, int pmax)

{

this.Min = pmin;

this.Max = pmax;

}

}

Stopwatch cl = new Stopwatch();

public Form1()

{

InitializeComponent();

}

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

{

cl.Reset();

OpenFileDialog win1 = new OpenFileDialog();

win1.InitialDirectory = "\\Mac/Home/Documents/Course\_2\bkIT/Lab\_4";

win1.Filter = "txt files (\*.txt)|\*.txt";//|All files (\*.\*)|\*.\*";

win1.FilterIndex = 2;

win1.RestoreDirectory = true;

if (win1.ShowDialog() == DialogResult.OK)

{

try

{

cl.Start();

string buf = File.ReadAllText(win1.FileName);

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

string[] buf2 = buf.Split();

foreach (string l in buf2)

{

if (!a.Contains(l))

a.Add(l);

}

a.Sort();

cl.Stop();

label3.Text = win1.FileName;

label1.Text = "Opening file, reading and sorting array(ms): " + cl.ElapsedMilliseconds.ToString();

addToListBox(a);

}

catch (Exception ex)

{

MessageBox.Show("ERROR! Couldn't read file from disk!\n Original error: " + ex.Message);

}

}

}

void addToListBox(List<string> a)

{

listBox1.BeginUpdate();

foreach (string l in a)

{

listBox1.Items.Add(l);

}

listBox1.EndUpdate();

}

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

{

cl.Reset();

cl.Start();

listBox1.SelectedIndex = listBox1.FindStringExact(textBox1.Text);

cl.Stop();

label2.Text = "Searching in ListBox(ms): " + cl.ElapsedMilliseconds.ToString();

LevFormlist(textBox1.Text);

}

public static int LevDist(string string1, string string2)

{

if (string1 == null) throw new ArgumentNullException("string1");

if (string2 == null) throw new ArgumentNullException("string2");

int diff;

int[,] m = new int[string1.Length + 1, string2.Length + 1];

for (int i = 0; i <= string1.Length; i++) { m[i, 0] = i; }

for (int j = 0; j <= string2.Length; j++) { m[0, j] = j; }

for (int i = 1; i <= string1.Length; i++)

{

for (int j = 1; j <= string2.Length; j++)

{

diff = (string1[i - 1] == string2[j - 1]) ? 0 : 1;

m[i, j] = Math.Min(Math.Min(m[i - 1, j] + 1,

m[i, j - 1] + 1),

m[i - 1, j - 1] + diff);

}

}

return m[string1.Length, string2.Length];

}

void LevFormlist(string s)

{

cl.Reset();

int p = 0;

bool f = int.TryParse(textBox2.Text, out p);

if (!f || p < 0)

{

throw new ArgumentException("Invalid Levenshtain parametr!");

}

listBox2.BeginUpdate();

listBox2.Items.Clear();

//

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

foreach (string l in listBox1.Items)

TempList.Add(l);

int count = int.Parse(ThreadCBox.Text);

List<ParallelSearchResult> Result = new List<ParallelSearchResult>(); //Результирующий список

List<MinMax> arrayDivList = DivideSubArrays(0, TempList.Count, count);

cl.Start();

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

for (int i = 0; i < count; i++) //Запуск потоков

{

List<string> tempTaskList = TempList.GetRange(arrayDivList[i].Min, arrayDivList[i].Max - arrayDivList[i].Min); //Создание временного списка, чтобы потоки не работали параллельно с одной коллекцией

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

//Метод, который будет выполняться в потоке

ArrayThreadTask,

//Параметры потока

new ParallelSearchThreadParam()

{

tempList = tempTaskList,

maxDist = p,

ThreadNum = i,

wordPattern = s

});

//Запуск потока

tasks[i].Start();

}

Task.WaitAll(tasks);

cl.Stop();

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

{

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

}

foreach (ParallelSearchResult x in Result)

{

listBox2.Items.Add(x.toString());

}

listBox2.EndUpdate();

label5.Text = " Making Levenshtein Distanse list(ms): " + cl.ElapsedMilliseconds;

}

public static List<MinMax> DivideSubArrays(int beginIndex, int endIndex, int subArraysCount)

{

List<MinMax> result = new List<MinMax>();

if ((endIndex - beginIndex) <= subArraysCount)

{

result.Add(new MinMax(0, (endIndex - beginIndex)));

}

else

{

int delta = (endIndex - beginIndex) / subArraysCount;

int currentBegin = beginIndex;

while ((endIndex - currentBegin) >= 2 \* delta)

{

result.Add(

new MinMax(currentBegin, currentBegin + delta));

currentBegin += delta;

}

result.Add(new MinMax(currentBegin, endIndex));

}

return result;

}

public static List<ParallelSearchResult> ArrayThreadTask(object paramObj)

{

/// Выполняется в параллельном потоке для поиска строк

ParallelSearchThreadParam param = (ParallelSearchThreadParam)paramObj;

string wordUpper = param.wordPattern.Trim().ToUpper(); //Слово для поиска в верхнем регистре

List<ParallelSearchResult> Result = new List<ParallelSearchResult>();//Результаты поиска в одном потоке

foreach (string str in param.tempList) //Перебор всех слов во временном списке данного потока

{

int dist = LevDist(str.ToUpper(), wordUpper);//Вычисление расстояния Дамерау-Левенштейна

if (dist <= param.maxDist)

{

ParallelSearchResult temp = new ParallelSearchResult()

{

word = str,

dist = dist,

ThreadNum = param.ThreadNum

};

Result.Add(temp);

}

}

return Result;

}

class ParallelSearchThreadParam

{

// Параметры которые передаются в поток

// для параллельного поиска

public List<string> tempList { get; set; } // Массив для поиска

public string wordPattern { get; set; } // Слово для поиска>

public int maxDist { get; set; } // Максимальное расстояние для нечеткого поиска

public int ThreadNum { get; set; } // Номер потока

}

private void button1\_Click\_1(object sender, EventArgs e)

{

OpenFileDialog win1 = new OpenFileDialog();

win1.InitialDirectory = "\\Mac/Home/Documents/Course\_2\bkIT/Lab\_4/Source";

win1.Filter = "txt files (\*.txt)|\*.txt";//|All files (\*.\*)|\*.\*";

win1.FilterIndex = 2;

win1.RestoreDirectory = true;

try

{

cl.Start();

string buf = null;

buf = label1.Text + "\n";

if (label2.Text != null)

{

buf = buf + label2.Text + " (searched item: " + listBox1.GetItemText(listBox1.SelectedIndex) + ")"+'\n' +

"Lev list (word\_LevDist\_Thread):" + '\n';

foreach (string l in listBox2.Items)

{

buf = buf + l + '\n';

}

}

else

{

buf = buf + "None searches made" + '\n';

}

SaveFileDialog fd = new SaveFileDialog();

string TempReportFileName = "Report\_" + DateTime.Now.ToString("dd\_MM\_yyyy\_hhmmss"); //Имя файла отчета

fd.FileName = TempReportFileName;

fd.DefaultExt = ".txt";

fd.Filter = "txt files|\*.txt";

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

{

File.AppendAllText(fd.FileName, buf);

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

}

}

catch (Exception ex)

{

MessageBox.Show("ERROR! Couldn't read file from disk!\n Original error: " + ex.Message);

}

}

}

}

### 3. Пример работы программы





