**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Базовые компоненты интернет-технологий»

Отчёт по домашнему заданию.

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-35б |  | преподаватель каф. ИУ5 |
| Зубарева Антонина |  |  |
| Подпись и дата: |  | Подпись и дата: |

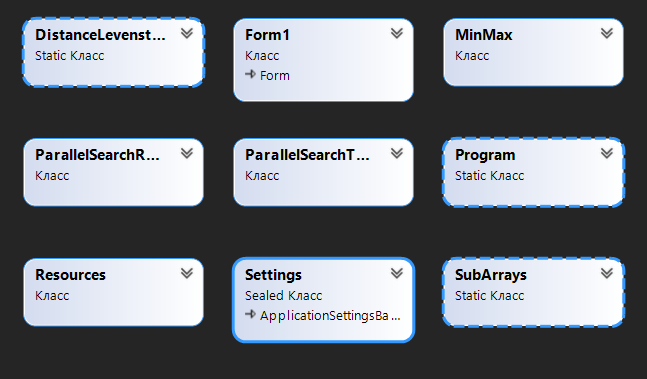
Москва, 2020 г.

# Задание.

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#».

# Диаграмма классов



# Текст программы

# using System;

# using System.Collections.Generic;

# using System.ComponentModel;

# using System.Data;

# using System.Diagnostics;

# using System.Drawing;

# using System.IO;

# using System.Linq;

# using System.Text;

# using System.Threading.Tasks;

# using System.Windows.Forms;

# namespace lab4

# {

# public partial class Form1 : Form

# {

# ///<summary>

# ///Список слов

# ///</summary>

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

# public Form1()

# {

# InitializeComponent();

# }

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

# {

# OpenFileDialog fd = new OpenFileDialog();

# fd.Filter = "текстовые файлы|\*.txt";

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

# {

# Stopwatch t\_load = new Stopwatch();

# t\_load.Start();

# // Чтение файла в виде одной строки

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

# //Разделительные символы для разбиения полученной строки

# 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\_load.Stop();

# this.textBoxFileReadTime.Text = t\_load.Elapsed.ToString();

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

# MessageBox.Show("Файл прочитан");

# }

# else {

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

# }

# }

# private void textBoxFileReadTime\_TextChanged(object sender, EventArgs e)

# {

# }

# private void buttonSearch\_Click(object sender, EventArgs e)

# {

# //Слово для поиска

# string 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\_search = new Stopwatch();

# t\_search.Start();

# foreach (string str in list)

# {

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

# {

# tempList.Add(str);

# }

# }

# t\_search.Stop();

# this.textBoxSearchTime.Text = t\_search.Elapsed.ToString();

# this.listBoxResult.BeginUpdate();

# //Очистка списка

# this.listBoxResult.Items.Clear();

# //Вывод результатов поиска

# 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 buttonParallelSearch\_Click(object sender, EventArgs e)

# {

# //Слово для поиска

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

# //Если слово для поиска не пусто

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

# {

# //максимальное расстояние

# int maxDist;

# if (!int.TryParse(this.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;

# }

# //количество потоков

# int ThreadCount;

# if (!int.TryParse(this.textBoxThreadCount.Text.Trim(), out ThreadCount))

# {

# MessageBox.Show("Необходимо указать количество потоков");

# return;

# }

# //время поиска

# Stopwatch timer = new Stopwatch();

# timer.Start();

# //-------------------------------------------------

# // Начало параллельного поиска

# //-------------------------------------------------

# //Результирующий список

# List<ParallelSearchResult> Result = new List<ParallelSearchResult>();

# //Деление списка на фрагменты для параллельного запуска в потоках

# List<MinMax> arrayDivList = SubArrays.DivideSubArrays(0, list.Count, ThreadCount);

# int count = arrayDivList.Count;

# //Количество потоков соответствует количеству фрагментов массива

# Task<List<ParallelSearchResult>>[] tasks = new Task<List<ParallelSearchResult>>[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(arrayDivList[i].Min, arrayDivList[i].Max - arrayDivList[i].Min);

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

# //Метод, который будет выполняться в потоке

# ArrayThreadTask,

# //Параметры потока

# new ParallelSearchThreadParam()

# {

# tempList = tempTaskList,

# maxDist = maxDist,

# ThreadNum = i,

# wordPattern = word

# });

# //Запуск потока

# tasks[i].Start();

# }

# Task.WaitAll(tasks);

# timer.Stop();

# //Объединение результатов

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

# {

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

# }

# //-------------------------------------------------

# // Завершение параллельного поиска

# //-------------------------------------------------

# timer.Stop();

# //Вывод результатов

# //Время поиска

# // this.textBoxApproxTime.Text = timer.Elapsed.ToString();

# //Вычисленное количество потоков

# this.textBoxThreadCountAll.Text = count.ToString();

# //Начало обновления списка результатов

# this.listBoxParallelResult.BeginUpdate();

# //Очистка списка

# this.listBoxParallelResult.Items.Clear();

# //Вывод результатов поиска

# foreach (var x in Result)

# {

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

# this.listBoxParallelResult.Items.Add(temp);

# }

# //Окончание обновления списка результатов

# this.listBoxParallelResult.EndUpdate();

# }

# else

# {

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

# }

# }

# /// <summary>

# /// Выполняется в параллельном потоке для поиска строк

# /// </summary>

# public static List<ParallelSearchResult> ArrayThreadTask(object paramObj)

# {

# ParallelSearchThreadParam param = (ParallelSearchThreadParam)paramObj;

# //Слово для поиска в верхнем регистре

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

# //Результаты поиска в одном потоке

# List<ParallelSearchResult> Result = new List<ParallelSearchResult>();

# //Перебор всех слов во временном списке данного потока

# foreach (string str in param.tempList)

# {

# //Вычисление расстояния Дамерау-Левенштейна

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

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

# if (dist <= param.maxDist)

# {

# ParallelSearchResult temp = new ParallelSearchResult()

# {

# word = str,

# dist = dist,

# ThreadNum = param.ThreadNum

# };

# Result.Add(temp);

# }

# }

# return Result;

# }

# private void buttonSaveReport\_Click(object sender, EventArgs e)

# {

# //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 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("<tr>");

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

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

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

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

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

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

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

# foreach (var x in this.listBoxParallelResult.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);

# }

# }

# }

# }

# Результаты

