Московский государственный технический университет

им. Н.Э. Баумана

Пояснительная записка к домашней работе

|  |  |
| --- | --- |
|  | Выполнила:  Персиянова В.  ИУ5-33 |

г. Москва

2017 г.

**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).

4. Реализуйте функцию записи результатов поиска в файл отчета. Файл отчета создается в формате .txt или .html.

**Текст программы на языке C#.**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

using System.Windows.Forms;

using DistanceLibrary;

using System.Threading.Tasks;

using System.Text;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public class ParallelSearchResult

{

public string word;

public int dist;

public int threadNum;

public ParallelSearchResult(string w, int d, int t)

{

word = w;

dist = d;

threadNum = t;

}

}

public class ParallelSearchThreadParam

{

public List<string> wordsList;

public string wordToFind;

public int maxDist;

public int threadNum;

public ParallelSearchThreadParam(List<string> wl, string wf, int m, int t)

{

wordsList = wl;

wordToFind = wf;

maxDist = m;

threadNum = t;

}

}

public class MinMax

{

public int min;

public int max;

public MinMax(int i, int j)

{

min = i;

max = j;

}

}

public List<MinMax> divideSubArrays(int wordsCount, int ThreadCount)

{

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

int countInThread = (int)Math.Ceiling(wordsCount / (double)ThreadCount);

int i = 0;

while (i < wordsCount)

{

MinMax mt = new MinMax(i, i+countInThread);

if (mt.max > wordsCount)

{

mt.max = wordsCount;

}

i += countInThread;

result.Add(mt);

}

return result;

}

public static List<ParallelSearchResult> arrayThreadTask(object p)

{

ParallelSearchThreadParam param = (ParallelSearchThreadParam)p;

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

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

foreach (string str in param.wordsList)

{

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

if (dist <= param.maxDist)

{

ParallelSearchResult t = new ParallelSearchResult(str, dist, param.threadNum);

result.Add(t);

}

}

return result;

}

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

private void Button\_fileLoad1(object sender, EventArgs e)

{

OpenFileDialog file = new OpenFileDialog();

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

{

Stopwatch time = new Stopwatch();

time.Start();

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

textBox1.Text = text;

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

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

foreach (string word in textArray)

{

string trimmedWord = word.Trim();

if (!words.Contains(trimmedWord))

{

words.Add(trimmedWord);

}

}

time.Stop();

label\_time.Text = time.Elapsed.ToString();

}

else

{

MessageBox.Show("Что-то не так!");

}

}

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

{

}

private void button\_findWord\_Click(object sender, EventArgs e)

{

string wordToFind = textBox\_wordToFind.Text.Trim();

textBox\_wordToFind.Text = wordToFind;

int maxDistance = Convert.ToInt32(textBox\_maxDistance.Text);

if (!string.IsNullOrEmpty(wordToFind) && words.Count > 0)

{

string wordToFindUpper = wordToFind.ToUpper();

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

int threadCount;

if (!int.TryParse(textBox2.Text.Trim(), out threadCount))

{

threadCount = 1;

}

Stopwatch time = new Stopwatch();

time.Start();

List<ParallelSearchResult> results = new List<ParallelSearchResult>();

List<MinMax> arrayDivList = divideSubArrays(words.Count, threadCount);

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

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

{

List<string> tempTaskList = words.GetRange(arrayDivList[i].min, arrayDivList[i].max - arrayDivList[i].min);

tasks[i] = new Task<List<ParallelSearchResult>>(arrayThreadTask, new ParallelSearchThreadParam(tempTaskList, wordToFindUpper, maxDistance, i));

tasks[i].Start();

}

Task.WaitAll(tasks);

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

{

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

}

time.Stop();

labelTimeFind.Text = time.Elapsed.ToString();

listBox.BeginUpdate();

listBox.Items.Clear();

foreach (var x in results)

{

string temp = x.word + " (расстояние - " + x.dist + ", поток - " + x.threadNum + ")";

listBox.Items.Add(temp);

}

listBox.EndUpdate();

}

else

{

MessageBox.Show("Что-то не то!");

}

}

private void label1\_Click(object sender, EventArgs e)

{

}

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

{

string TempReportFileName = "Отчет\_" + DateTime.Now.ToString("dd\_MM\_yyyy\_hhmmss");

SaveFileDialog fd = new SaveFileDialog();

fd.FileName = TempReportFileName;

fd.DefaultExt = ".html";

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

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

{

string ReportFileName = fd.FileName;

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.label\_time.Text + "</td>");

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

b.AppendLine("<tr>");

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

b.AppendLine("<tr>");

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

b.AppendLine("<td>" + this.textBox\_wordToFind.Text + "</td>");

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

b.AppendLine("<tr>");

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

b.AppendLine("<td>" + this.textBox\_maxDistance.Text + "</td>");

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

b.AppendLine("<tr>");

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

b.AppendLine("<td>" + this.labelTimeFind.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.listBox.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>");

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

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

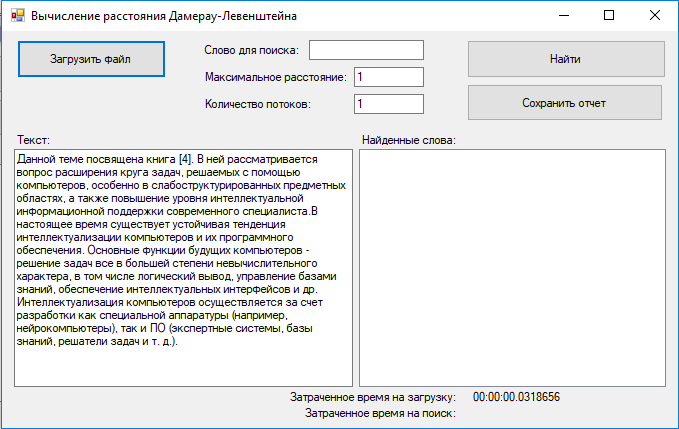
}

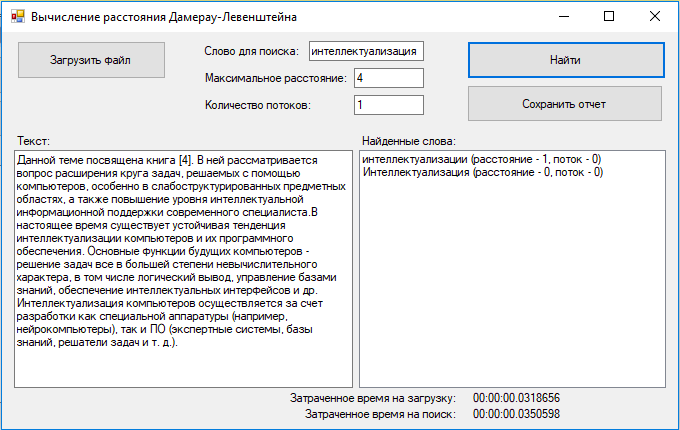
}

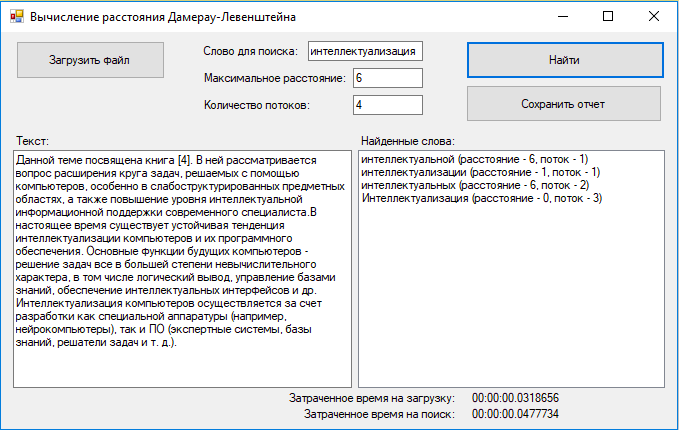
}

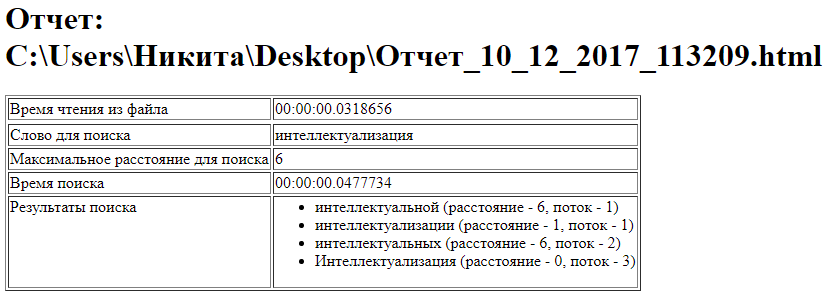
}

**Результаты выполнения программы:**

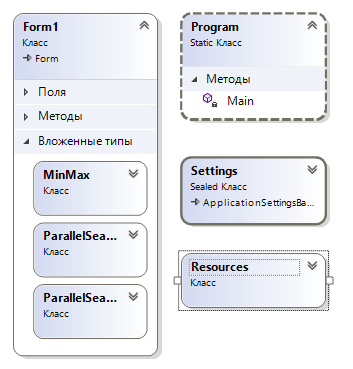
****

****

****

****

**Диаграмма классов:**

****