**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по домашнему заданию

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Выполнил: |  |  |  |  |
| студент группы ИУ5-33Б |  |  |  |  | |
|  |  |  |  |  |
| Тарновский Даниил |  |  |  |  |

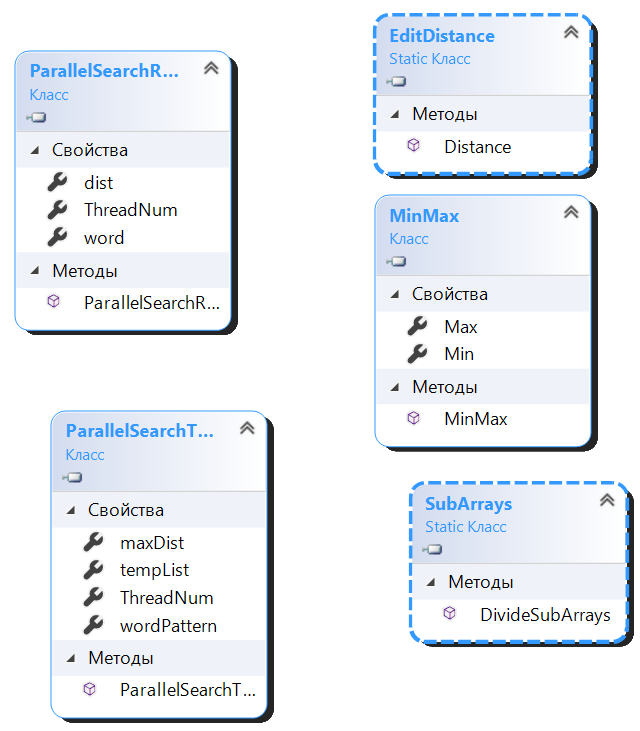
Москва, 2020 г.

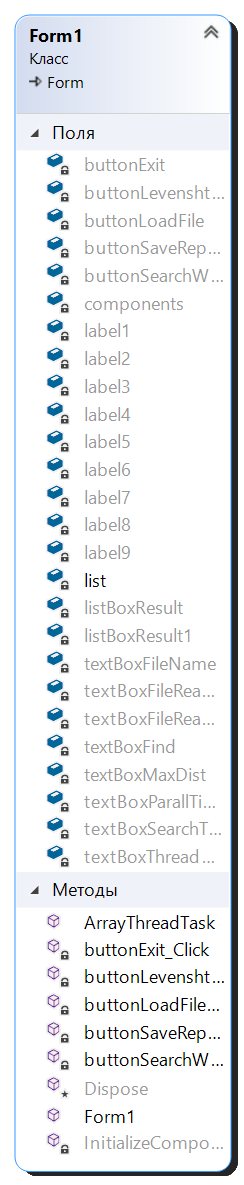
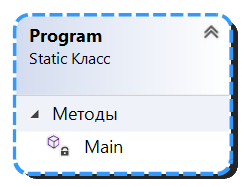
**Домашнее задание**

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:**

Program.cs

using System;

//using System.Collections.Generic;

//using System.Linq;

//using System.Threading.Tasks;

using System.Windows.Forms;

using MyLib;

namespace Distant

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Form1.cs

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Diagnostics;

using System.Threading.Tasks;

using MyLib;

namespace Distant

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

/// <summary>

/// Список слов

/// </summary>

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

// Считывание файла

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");

}

}

// Поиск слова

private void buttonSearchWord\_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 = new Stopwatch();

t.Start();

foreach (string str in list)

{

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

{

tempList.Add(str);

}

}

t.Stop();

this.textBoxSearchTime.Text = t.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 buttonLevenshteinDist\_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)

{

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;

}

int ThreadCount;

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

{

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

return;

}

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 паралельного поиска

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

//Начало обновления списка результатов

this.listBoxResult1.BeginUpdate();

//Очистка списка

this.listBoxResult1.Items.Clear();

//Вывод результатов поиска

foreach (var x in Result)

{

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

this.listBoxResult1.Items.Add(temp);

}

//Окончание обновления списка результатов

this.listBoxResult1.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 = EditDistance.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;

}

// Отчет

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

{

//Имя файла отчета

string TempReportFileName = "Отчет\_" + 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>Время поиска</td>");

b.AppendLine("<td>" + this.textBoxSearchTime.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 параллельного поиска</td>");

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

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

b.AppendLine("<tr>");

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

b.AppendLine("<td>" + this.textBoxParallTime.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.listBoxResult1.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);

}

}

// Выход

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

{

this.Close();

}

}

}

Form1.Designer.cs

namespace Distant

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.buttonLoadFile = new System.Windows.Forms.Button();

this.buttonExit = new System.Windows.Forms.Button();

this.buttonLevenshteinDist = new System.Windows.Forms.Button();

this.buttonSearchWord = new System.Windows.Forms.Button();

this.buttonSaveReport = new System.Windows.Forms.Button();

this.textBoxFileReadTime = new System.Windows.Forms.TextBox();

this.textBoxParallTime = new System.Windows.Forms.TextBox();

this.textBoxFileName = new System.Windows.Forms.TextBox();

this.textBoxFileReadCount = new System.Windows.Forms.TextBox();

this.textBoxFind = new System.Windows.Forms.TextBox();

this.textBoxMaxDist = new System.Windows.Forms.TextBox();

this.textBoxSearchTime = new System.Windows.Forms.TextBox();

this.textBoxThreadCount = new System.Windows.Forms.TextBox();

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.label3 = new System.Windows.Forms.Label();

this.label4 = new System.Windows.Forms.Label();

this.label5 = new System.Windows.Forms.Label();

this.label8 = new System.Windows.Forms.Label();

this.label6 = new System.Windows.Forms.Label();

this.label7 = new System.Windows.Forms.Label();

this.label9 = new System.Windows.Forms.Label();

this.listBoxResult = new System.Windows.Forms.ListBox();

this.listBoxResult1 = new System.Windows.Forms.ListBox();

this.SuspendLayout();

//

// buttonLoadFile

//

this.buttonLoadFile.Location = new System.Drawing.Point(16, 31);

this.buttonLoadFile.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.buttonLoadFile.Name = "buttonLoadFile";

this.buttonLoadFile.Size = new System.Drawing.Size(177, 35);

this.buttonLoadFile.TabIndex = 0;

this.buttonLoadFile.Text = "Выбрать файл";

this.buttonLoadFile.UseVisualStyleBackColor = true;

this.buttonLoadFile.Click += new System.EventHandler(this.buttonLoadFile\_Click);

//

// buttonExit

//

this.buttonExit.Location = new System.Drawing.Point(1442, 560);

this.buttonExit.Name = "buttonExit";

this.buttonExit.Size = new System.Drawing.Size(127, 39);

this.buttonExit.TabIndex = 14;

this.buttonExit.Text = "Выход";

this.buttonExit.UseVisualStyleBackColor = true;

this.buttonExit.Click += new System.EventHandler(this.buttonExit\_Click);

//

// buttonLevenshteinDist

//

this.buttonLevenshteinDist.Location = new System.Drawing.Point(914, 212);

this.buttonLevenshteinDist.Name = "buttonLevenshteinDist";

this.buttonLevenshteinDist.Size = new System.Drawing.Size(148, 62);

this.buttonLevenshteinDist.TabIndex = 9;

this.buttonLevenshteinDist.Text = "Параллельный поиск";

this.buttonLevenshteinDist.UseVisualStyleBackColor = true;

this.buttonLevenshteinDist.Click += new System.EventHandler(this.buttonLevenshteinDist\_Click);

//

// buttonSearchWord

//

this.buttonSearchWord.Location = new System.Drawing.Point(12, 149);

this.buttonSearchWord.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.buttonSearchWord.Name = "buttonSearchWord";

this.buttonSearchWord.Size = new System.Drawing.Size(177, 35);

this.buttonSearchWord.TabIndex = 2;

this.buttonSearchWord.Text = "Поиск";

this.buttonSearchWord.UseVisualStyleBackColor = true;

this.buttonSearchWord.Click += new System.EventHandler(this.buttonSearchWord\_Click);

//

// buttonSaveReport

//

this.buttonSaveReport.Location = new System.Drawing.Point(11, 554);

this.buttonSaveReport.Name = "buttonSaveReport";

this.buttonSaveReport.Size = new System.Drawing.Size(182, 39);

this.buttonSaveReport.TabIndex = 15;

this.buttonSaveReport.Text = "Сохранить отчет";

this.buttonSaveReport.UseVisualStyleBackColor = true;

this.buttonSaveReport.Click += new System.EventHandler(this.buttonSaveReport\_Click);

//

// textBoxFileReadTime

//

this.textBoxFileReadTime.Location = new System.Drawing.Point(592, 271);

this.textBoxFileReadTime.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.textBoxFileReadTime.Name = "textBoxFileReadTime";

this.textBoxFileReadTime.ReadOnly = true;

this.textBoxFileReadTime.Size = new System.Drawing.Size(199, 26);

this.textBoxFileReadTime.TabIndex = 14;

//

// textBoxParallTime

//

this.textBoxParallTime.Location = new System.Drawing.Point(592, 510);

this.textBoxParallTime.Name = "textBoxParallTime";

this.textBoxParallTime.ReadOnly = true;

this.textBoxParallTime.Size = new System.Drawing.Size(199, 26);

this.textBoxParallTime.TabIndex = 11;

//

// textBoxFileReadCount

//

this.textBoxFileReadCount.Location = new System.Drawing.Point(592, 347);

this.textBoxFileReadCount.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.textBoxFileReadCount.Name = "textBoxFileReadCount";

this.textBoxFileReadCount.ReadOnly = true;

this.textBoxFileReadCount.Size = new System.Drawing.Size(199, 26);

this.textBoxFileReadCount.TabIndex = 4;

//

// textBoxFind

//

this.textBoxFind.Location = new System.Drawing.Point(274, 97);

this.textBoxFind.Name = "textBoxFind";

this.textBoxFind.Size = new System.Drawing.Size(561, 26);

this.textBoxFind.TabIndex = 1;

//

// textBoxMaxDist

//

this.textBoxMaxDist.Location = new System.Drawing.Point(1349, 131);

this.textBoxMaxDist.Name = "textBoxMaxDist";

this.textBoxMaxDist.Size = new System.Drawing.Size(82, 26);

this.textBoxMaxDist.TabIndex = 21;

//

// textBoxSearchTime

//

this.textBoxSearchTime.Location = new System.Drawing.Point(592, 420);

this.textBoxSearchTime.Name = "textBoxSearchTime";

this.textBoxSearchTime.ReadOnly = true;

this.textBoxSearchTime.Size = new System.Drawing.Size(199, 26);

this.textBoxSearchTime.TabIndex = 10;

//

// textBoxThreadCount

//

this.textBoxThreadCount.Location = new System.Drawing.Point(1363, 175);

this.textBoxThreadCount.Name = "textBoxThreadCount";

this.textBoxThreadCount.Size = new System.Drawing.Size(82, 26);

this.textBoxThreadCount.TabIndex = 19;

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(530, 233);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(197, 20);

this.label1.TabIndex = 6;

this.label1.Text = "Время чтения из файла:";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(530, 302);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(305, 20);

this.label2.TabIndex = 7;

this.label2.Text = "Количество уникальных слов в файле:";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(12, 103);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(245, 20);

this.label3.TabIndex = 8;

this.label3.Text = "Введите значение для поиска:";

//

// label4

//

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(530, 378);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(118, 20);

this.label4.TabIndex = 9;

this.label4.Text = "Время поиска:";

//

// label5

//

this.label5.AutoSize = true;

this.label5.Location = new System.Drawing.Point(1106, 210);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(149, 20);

this.label5.TabIndex = 12;

this.label5.Text = "Результат поиска:";

//

// label8

//

this.label8.AutoSize = true;

this.label8.Location = new System.Drawing.Point(884, 134);

this.label8.Name = "label8";

this.label8.Size = new System.Drawing.Size(439, 20);

this.label8.TabIndex = 18;

this.label8.Text = "Максимальное расстояние для параллельногор поиска:";

//

// label6

//

this.label6.AutoSize = true;

this.label6.Location = new System.Drawing.Point(530, 474);

this.label6.Name = "label6";

this.label6.Size = new System.Drawing.Size(240, 20);

this.label6.TabIndex = 20;

this.label6.Text = "Время параллельного поиска:";

//

// label7

//

this.label7.AutoSize = true;

this.label7.Location = new System.Drawing.Point(957, 164);

this.label7.Name = "label7";

this.label7.Size = new System.Drawing.Size(170, 20);

this.label7.TabIndex = 18;

this.label7.Text = "Количество потоков:";

//

// label9

//

this.label9.AutoSize = true;

this.label9.Location = new System.Drawing.Point(12, 210);

this.label9.Name = "label9";

this.label9.Size = new System.Drawing.Size(149, 20);

this.label9.TabIndex = 12;

this.label9.Text = "Результат поиска:";

//

// listBoxResult

//

this.listBoxResult.FormattingEnabled = true;

this.listBoxResult.ItemHeight = 20;

this.listBoxResult.Location = new System.Drawing.Point(4, 233);

this.listBoxResult.Name = "listBoxResult";

this.listBoxResult.Size = new System.Drawing.Size(484, 284);

this.listBoxResult.TabIndex = 11;

//

// listBoxResult1

//

this.listBoxResult1.FormattingEnabled = true;

this.listBoxResult1.ItemHeight = 20;

this.listBoxResult1.Location = new System.Drawing.Point(1110, 243);

this.listBoxResult1.Name = "listBoxResult1";

this.listBoxResult1.Size = new System.Drawing.Size(415, 284);

this.listBoxResult1.TabIndex = 11;

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 20F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(1924, 607);

this.Controls.Add(this.textBoxFileReadCount);

this.Controls.Add(this.textBoxFileReadTime);

this.Controls.Add(this.textBoxParallTime);

this.Controls.Add(this.textBoxThreadCount);

this.Controls.Add(this.buttonLoadFile);

this.Controls.Add(this.buttonSearchWord);

this.Controls.Add(this.buttonExit);

this.Controls.Add(this.buttonLevenshteinDist);

this.Controls.Add(this.buttonSaveReport);

this.Controls.Add(this.label1);

this.Controls.Add(this.label2);

this.Controls.Add(this.label3);

this.Controls.Add(this.label4);

this.Controls.Add(this.label5);

this.Controls.Add(this.label6);

this.Controls.Add(this.label7);

this.Controls.Add(this.label8);

this.Controls.Add(this.label9);

this.Controls.Add(this.textBoxFind);

this.Controls.Add(this.textBoxMaxDist);

this.Controls.Add(this.textBoxFileName);

this.Controls.Add(this.textBoxSearchTime);

this.Controls.Add(this.listBoxResult);

this.Controls.Add(this.listBoxResult1);

this.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);

this.Name = "Form1";

this.Text = "Поиск в файле";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button buttonLoadFile;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.TextBox textBoxFileReadTime;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.Label label6;

private System.Windows.Forms.Label label7;

private System.Windows.Forms.Label label8;

private System.Windows.Forms.Label label9;

private System.Windows.Forms.TextBox textBoxFileReadCount;

private System.Windows.Forms.TextBox textBoxFileName;

private System.Windows.Forms.TextBox textBoxMaxDist;

private System.Windows.Forms.TextBox textBoxThreadCount;

private System.Windows.Forms.Button buttonSearchWord;

private System.Windows.Forms.TextBox textBoxFind;

private System.Windows.Forms.TextBox textBoxSearchTime;

private System.Windows.Forms.TextBox textBoxParallTime;

private System.Windows.Forms.ListBox listBoxResult;

private System.Windows.Forms.ListBox listBoxResult1;

private System.Windows.Forms.Button buttonExit;

private System.Windows.Forms.Button buttonSaveReport;

private System.Windows.Forms.Button buttonLevenshteinDist;

}

}

EditDistance.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MyLib

{

public static class EditDistance

{

/// <summary>

/// Вычисление расстояния Дамерау-Левенштейна

/// </summary>

public static int Distance(string str1Param, string str2Param)

{

if ((str1Param == null) || (str2Param == null)) return -1;

int str1Len = str1Param.Length;

int str2Len = str2Param.Length;

//Если хотя бы одна строка пустая, возвращается длина другой строки

if ((str1Len == 0) && (str2Len == 0)) return 0;

if (str1Len == 0) return str2Len;

if (str2Len == 0) return str1Len;

//Приведение строк к верхнему регистру

string str1 = str1Param.ToUpper();

string str2 = str2Param.ToUpper();

//Объявление матрицы

int[,] matrix = new int[str1Len + 1, str2Len + 1];

//Инициализация нулевой строки и нулевого столбца матрицы

for (int i = 0; i <= str1Len; i++) matrix[i, 0] = i;

for (int j = 0; j <= str2Len; j++) matrix[0, j] = j;

//Вычисление расстояния Дамерау-Левенштейна

for (int i = 1; i <= str1Len; i++)

{

for (int j = 1; j <= str2Len; j++)

{

//Эквивалентность символов, переменная symbEqual соответствует m(s1[i],s2[j])

int symbEqual = ((str1.Substring(i - 1, 1) == str2.Substring(j - 1, 1)) ? 0 : 1);

int ins = matrix[i, j - 1] + 1; //Добавление

int del = matrix[i - 1, j] + 1; //Удаление

int subst = matrix[i - 1, j - 1] + symbEqual; //Замена

//Элемент матрицы вычисляется как минимальный из трех случаев

matrix[i, j] = Math.Min(Math.Min(ins, del), subst);

//Дополнение Дамерау по перестановке соседних символов

if ((i > 1) && (j > 1) &&

(str1.Substring(i - 1, 1) == str2.Substring(j - 2, 1)) &&

(str1.Substring(i - 2, 1) == str2.Substring(j - 1, 1)))

{

matrix[i, j] = Math.Min(matrix[i, j], matrix[i - 2, j - 2] + symbEqual);

}

}

}

//Возвращается нижний правый элемент матрицы

return matrix[str1Len, str2Len];

}

}

}

MinMax.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MyLib

{

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;

}

}

}

ParallelSearchResult.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Distant

{

public class ParallelSearchResult

{

/// <summary>

/// Найденное слово

/// </summary>

public string word { get; set; }

/// <summary>

/// Расстояние

/// </summary>

public int dist { get; set; }

/// <summary>

/// Номер потока

/// </summary>

public int ThreadNum { get; set; }

}

}

ParallelSearchThreadParam.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Distant

{

public class ParallelSearchThreadParam

{

/// <summary>

/// Массив для поиска

/// </summary>

public List<string> tempList { get; set; }

/// <summary>

/// Слово для поиска

/// </summary>

public string wordPattern { get; set; }

/// <summary>

/// Максимальное расстояние для нечеткого поиска

/// </summary>

public int maxDist { get; set; }

/// <summary>

/// Номер потока

/// </summary>

public int ThreadNum { get; set; }

}

}

SubArrays.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MyLib

{

public static class SubArrays

{

/// <summary>

/// Деление массива на последовательности

/// </summary>

/// <param name="beginIndex">Начальный индекс массива</param>

/// <param name="endIndex">Конечный индекс массива</param>

/// <param name="subArraysCount">Требуемое количество подмассивов</param>

/// <returns>Список пар с индексами подмассивов</returns>

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

{

//Результирующий список пар с индексами подмассивов

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

//Если число элементов в массиве слишком мало для деления

//то возвращается массив целиком

if ((endIndex - beginIndex) <= subArraysCount)

{

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

}

else

{

//Размер подмассива

int delta = (endIndex - beginIndex) / subArraysCount;

//Начало отсчета

int currentBegin = beginIndex;

//Пока размер подмассива укладывается в оставшуюся последовательность

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

{

//Формируем подмассив на основе начала последовательности

result.Add(new MinMax(currentBegin, currentBegin + delta));

//Сдвигаем начало последовательности вперед на размер подмассива

currentBegin += delta;

}

//Оставшийся фрагмент массива

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

}

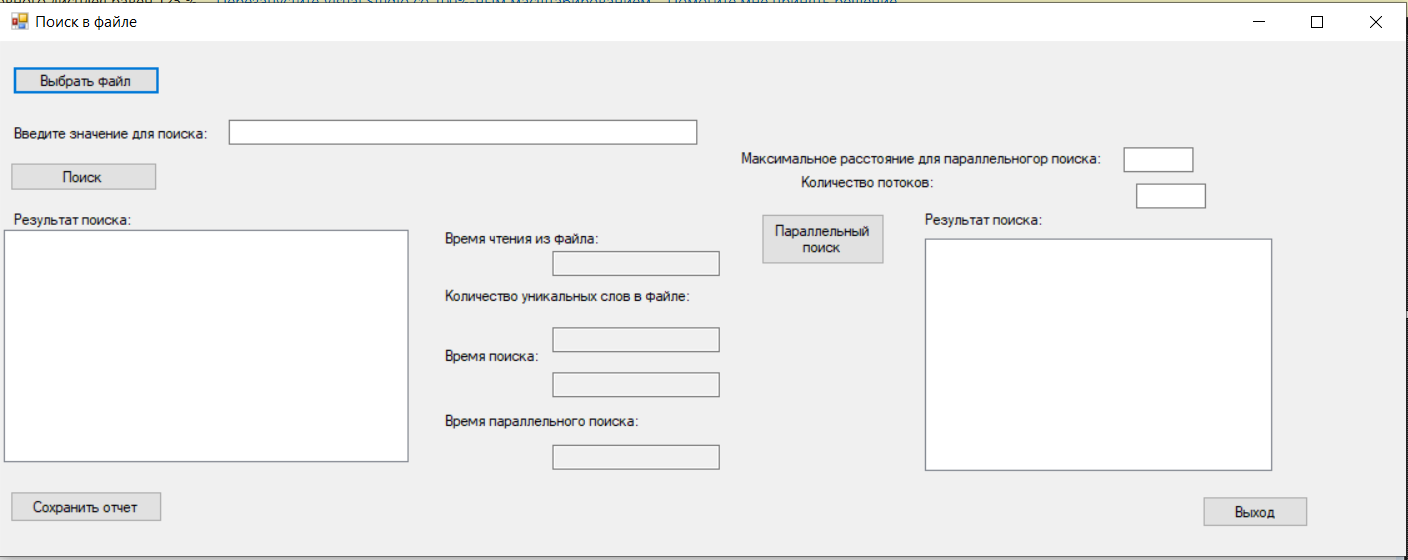
//Возврат списка результатов

return result;

}

}

}

****

**Примеры выполнения программы:**

