**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Пояснительная записка по выполнению

домашнего задания

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б |  | преподаватель каф. ИУ5 |
| Белова Е.О. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

г. Москва, 2018 г.

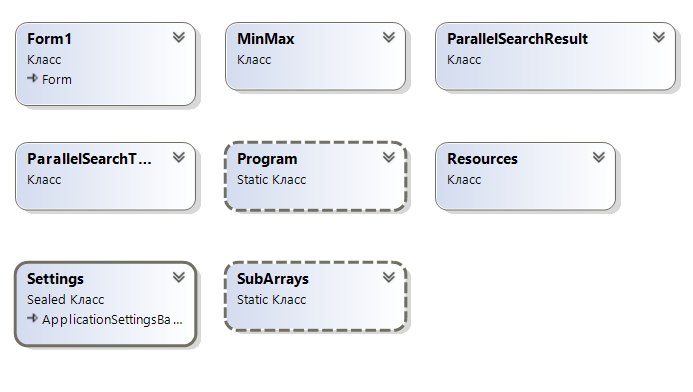
**Описание задания:**

Пример реализации ДЗ рассмотрен в учебном пособии, глава «Пример многопоточного поиска в текстовом файле с использованием технологии Windows Forms».

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.

**Диаграмма классов:**



**Текст программы:**

**Program.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace win\_form

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

**MinMax.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace win\_form

{

///<summary>

///Хранение минимального и максимального значений диапазона

///</summary>

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:**

namespace win\_form

{

///<summary>

///Результаты параллельного поиска

///</summary>

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.Collections.Generic;

namespace win\_form

{

///<summary>

///Параметры которые передаются в поток

///для параллельного поиска

///</summary>

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.Collections.Generic;

namespace win\_form

{

///<summary>

///Класс для деления массива на последовательности

///</summary>

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:**

