Московский государственный технический
университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Пояснительная записка по выполнению домашнего задания

Выполнил:
студент группы РТ5-31Б

Ходосов Михаил
Подпись и дата:

Проверил:
доцент каф. ИУ5

Гапанюк Ю.Е.
Подпись и дата:

Москва, 2020 г.

Описание задания

Разработать программу, реализующую многопоточный поиск в файле.

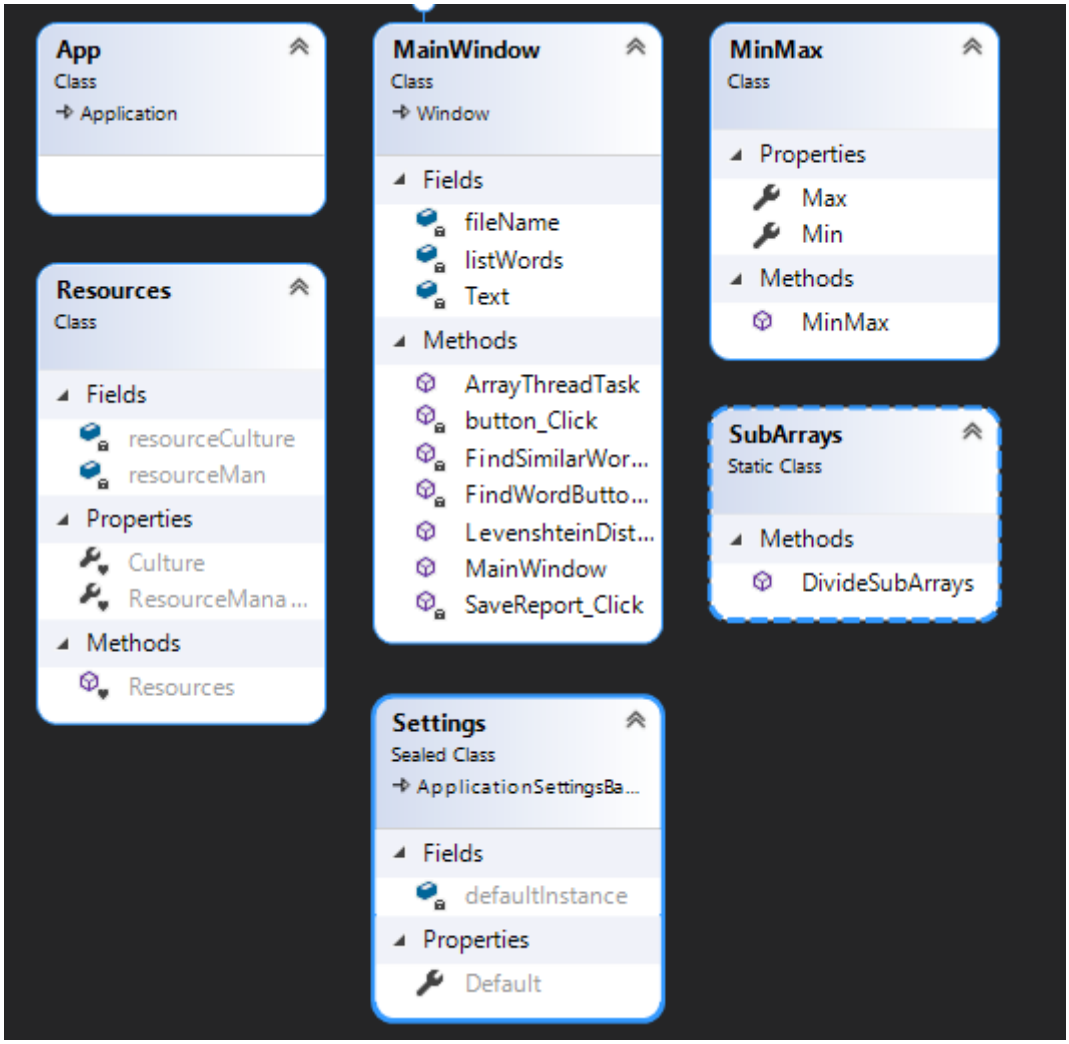
1. Программа должна быть разработана в виде приложения Windows Forms на языке C#. По желанию вместо Windows Forms возможно использование WPF.

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). 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

MainWindow.xaml

```
<Window x:Class="homework.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:homework"
        mc:Ignorable="d"
        Title="Домашнее задание" Height="450" Width="1200">
    <Grid Width="1200" Height="420">
        <StackPanel Orientation="Horizontal" Margin="0,0,0,0">
            <Button x:Name="OpenFileButton" Content="Чтение из файла"
HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="100"
Height="40" Click="button_Click"/>
            <Label x:Name="ReadingTimeLabel" VerticalAlignment="Top" Content="Время
чтения:" Height="30" Width="120" Margin="10, 5, 0, 0"/>
            <TextBox x:Name="ElapsedTime" Height="20" Margin="-120,30,0,0"
VerticalAlignment="Top" Width="120"/>
            <Label x:Name="SearchingTimeLabel" VerticalAlignment="Top" Content="Время
поиска:" Height="30" Width="120" Margin="210, 10, 0, 0"/>
            <TextBox x:Name="SearchingTime" Height="20" Margin="-185,40,-70,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>
            <Button x:Name="FindWordButton" Content="Найти слово" Height="20"
Margin="10,10,0,0" VerticalAlignment="Top" Width="140" Click="FindWordButton_Click"/>
            <TextBox x:Name="FindWordField" TextWrapping="Wrap" Width="140" Height="20"
Margin="-140,40,10,2" VerticalAlignment="Top"/>
            <Label x:Name="SampleWordLabel" VerticalAlignment="Top" Content="Слово для
поиска:" Height="30" Width="180" Margin="60, 20, 0, 0"/>
            <TextBox x:Name="SampleWord" Height="20" Margin="-175,50,0,0"
VerticalAlignment="Top" Width="180"/>
            <Button x:Name="SaveReport" Content="Сохранить отчет" Height="20" Margin="-
220,0,0,0" VerticalAlignment="Top" Width="140" Click="SaveReport_Click"/>
            <TextBlock x:Name="MaxDistLabel" VerticalAlignment="Top"
TextWrapping="WrapWithOverflow" Text="Максимальное расстояние:" Height="40" Width="90"
Margin="10, 5, 0, 0" HorizontalAlignment="Center"/>
            <TextBox x:Name="MaxDistText" Height="20" Margin="-165,50,-70,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="90"/>
            <TextBlock x:Name="MaxThreadCountLabel" VerticalAlignment="Top"
TextWrapping="WrapWithOverflow" Text="Количество потоков:" Height="40" Width="90"
Margin="10, 5, 0, 0" HorizontalAlignment="Center"/>
            <TextBox x:Name="MaxThreadCountText" Height="20" Margin="-165,50,-70,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="90"/>
            <Button x:Name="FindSimilarWords" Content="Найти похожие слова" Width="380"
Margin="-380, 80, 0, 0" VerticalAlignment="Top" Click="FindSimilarWords_Click"/>

        </StackPanel>
        <Label x:Name="PathFileLabel" Content="Путь к файлу:" Height="30" Width="120"
Margin="-1065, -300, 0, 0"/>
        <ListBox x:Name="Content" HorizontalAlignment="Left" Height="308"
Margin="10,100,0,0" VerticalAlignment="Top" Width="390"/>
        <TextBox x:Name="PathFile" HorizontalAlignment="Left" Height="20"
Margin="10,70,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="390"/>
        <ListBox x:Name="FoundWords" HorizontalAlignment="Left" Height="308"
Margin="455,100,10,0" VerticalAlignment="Top" Width="265"/>
        <ListBox x:Name="SimilarWordsBox" HorizontalAlignment="Left" Height="298"
Margin="795,110,0,0" VerticalAlignment="Top" Width="380"/>

    </Grid>
</Window>
```

MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Diagnostics;
using System.Threading;
using Parallel;

namespace homework
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        List<string> listWords = new List<string>();
        private string fileName;
        private string Text;
        public MainWindow()
        {
            InitializeComponent();
        }
        //Чтение из файла
        private void button_Click(object sender, RoutedEventArgs e)
        {
            Content.Items.Clear();
            listWords.Clear();
            fileName = "";
            PathFile.Text = "";
            Text = "";

            Microsoft.Win32.OpenFileDialog fileDialog = new
Microsoft.Win32.OpenFileDialog();
            fileDialog.Filter = "Только текстовые файлы|*.txt";
            if (fileDialog.ShowDialog() == true)
            {
                Stopwatch timeLoading = new Stopwatch();
                timeLoading.Start();

                fileName = fileDialog.FileName;
                PathFile.Text = fileName;

                Text = File.ReadAllText(fileName);

                string[] words = Text.Split(' ', ',', '.', '?', '!', '/', '|', '"', '\n',
'\t', '_', '-', '(', ')', '*', '{', '}', '[', ']');

                foreach (string word in words)
                {
                    if (!listWords.Contains(word))
                    {
                        listWords.Add(word);
```

```
                    }
                }

                timeLoading.Stop();
                ElapsedTime.Text = timeLoading.Elapsed.ToString();

                foreach (string word in listWords)
                {
                    Content.Items.Add(word);
                }

            }
        }
        //Поиск слов, для которых заданная строка является подстрокой
        private void FindWordButton_Click(object sender, RoutedEventArgs e)
        {
            FoundWords.Items.Clear();

            if (FindWordField.Text == null)
                return;

            Stopwatch timeSearching = new Stopwatch();
            timeSearching.Start();

            foreach (string word in listWords)
            {
                if (word.Contains(FindWordField.Text))
                {
                    FoundWords.Items.Add(word);
                }
            }

            timeSearching.Stop();
            SearchingTime.Text = timeSearching.Elapsed.ToString();
        }

        //Поиск похожих слов
        private void FindSimilarWords_Click(object sender, RoutedEventArgs e)
        {
            string sampleWord = SampleWord.Text.Trim();
            int maxDistance = Convert.ToInt32(MaxDistText.Text);
            int ThreadCount = Convert.ToInt32(MaxThreadCountText.Text);

            if (!string.IsNullOrWhiteSpace(sampleWord) && listWords.Count > 0)
            {
                //Результирующий список
                List<string> Result = new List<string>();

                //Деление списка на фрагменты для параллельного запуска в потоках
                List<MinMax> arrayDivList = SubArrays.DivideSubArrays(0, listWords.Count,
ThreadCount);
                int count = arrayDivList.Count;


                //Количество потоков соответствует количеству фрагментов массива
                Task<List<string>>[] tasks = new Task<List<string>>[count];

                //Запуск потоков
                for (int i = 0; i < count; i++)
                {
                    //Создание временного списка, чтобы потоки не работали параллельно с
одной коллекцией
                    List<string> tempTaskList = listWords.GetRange(arrayDivList[i].Min,
arrayDivList[i].Max - arrayDivList[i].Min);
```

```
                    tasks[i] = new Task<List<string>>(
                        //Метод, который будет выполняться в потоке
                        ArrayThreadTask,
                        //Параметры потока передаются в виде кортежа, чтобы не создавать
временный класс
                        new Tuple<List<String>, string,  int>(tempTaskList, sampleWord,
maxDistance));
                    //Запуск потока
                    tasks[i].Start();
                }

                //Ожидание завершения всех потоков
                Task.WaitAll(tasks);

                //Объединение результатов полученных из разных потоков
                for (int i = 0; i < count; i++)
                {
                    //Добавление результатов конкретного потока в общий массив
результатов
                    Result.AddRange(tasks[i].Result);
                }

                SimilarWordsBox.Items.Clear();
                foreach (string word in Result)
                {
                    SimilarWordsBox.Items.Add(word);
                }

            }
        }

        public static List<string> ArrayThreadTask(object paramObj)
        {
            //Получение параметров
            Tuple<List<string>, string, int> param = (Tuple<List<string>, string,
int>)paramObj;
            int listCount = param.Item1.Count;

            //Временный список для результата
            List<string> tempData = new List<string>();

            string word = param.Item2;

            //Перебор нужных элементов в списке данных
            for (int i = 0; i < listCount; i++)
            {
                string temp = param.Item1[i];
                int dist = LevenshteinDistance(word, temp);
                if (dist <= param.Item3)
                {
                    tempData.Add(temp);
                }
            }

            //Возврат массива данных
            return tempData;
        }

        //Поиск расстояния Левенштейна
        public static int LevenshteinDistance(string str1, string str2)
        {
            //Проверка на исключительные случаи

            if ((str1 == null && str2 == null) || (str1 == str2)) return 0;
```

```
            if (str1 == null || str2 == null) throw new ArgumentNullException("Одна из
строк пустая!\n");

            //Алгоритм Вагнера – Фишера

            int[,] matrix = new int[str1.Length + 1, str2.Length + 1];

            for (int i = 0; i <= str1.Length; i++)
            {
                matrix[i, 0] = i;
            }
            for (int j = 0; j <= str2.Length; j++)
            {
                matrix[0, j] = j;
            }

            for (int i = 1; i <= str1.Length; i++)
            {
                for (int j = 1; j <= str2.Length; j++)
                {
                    int d = 1;
                    if (str1[i - 1] == str2[j - 1]) d = 0;
                    matrix[i, j] = Math.Min(Math.Min(matrix[i - 1, j] + 1, matrix[i, j -
1] + 1), matrix[i - 1, j - 1] + d);

                }
            }
            return matrix[str1.Length, str2.Length];
        }

        private void SaveReport_Click(object sender, RoutedEventArgs e)
        {
            //Имя файла отчета
            string TempReportFileName = "Report_" +
DateTime.Now.ToString("dd_MM_yyyy_hhmmss");

            //Диалог сохранения файла отчета
            Microsoft.Win32.SaveFileDialog fd = new Microsoft.Win32.SaveFileDialog();
            fd.FileName = TempReportFileName;
            fd.DefaultExt = ".html";
            fd.Filter = "HTML Reports|*.html";

            if (fd.ShowDialog() == true)
            {
                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>" + ElapsedTime.Text + "</td>");
                b.AppendLine("</tr>");
```

```
                b.AppendLine("<tr>");
                b.AppendLine("<td>Слово для поиска</td>");
                b.AppendLine("<td>" + SampleWord.Text + "</td>");
                b.AppendLine("</tr>");

                b.AppendLine("<tr>");
                b.AppendLine("<td>Максимальное расстояние для нечеткого поиска</td>");
                b.AppendLine("<td>" + MaxDistText.Text + "</td>");
                b.AppendLine("</tr>");

                b.AppendLine("<tr>");
                b.AppendLine("<td>Время четкого поиска</td>");
                b.AppendLine("<td>" + SearchingTime.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.SimilarWordsBox.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);
            }
        }
    }
}
```

MinMax.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Parallel
{
    /// <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;
        }
    }
}
```

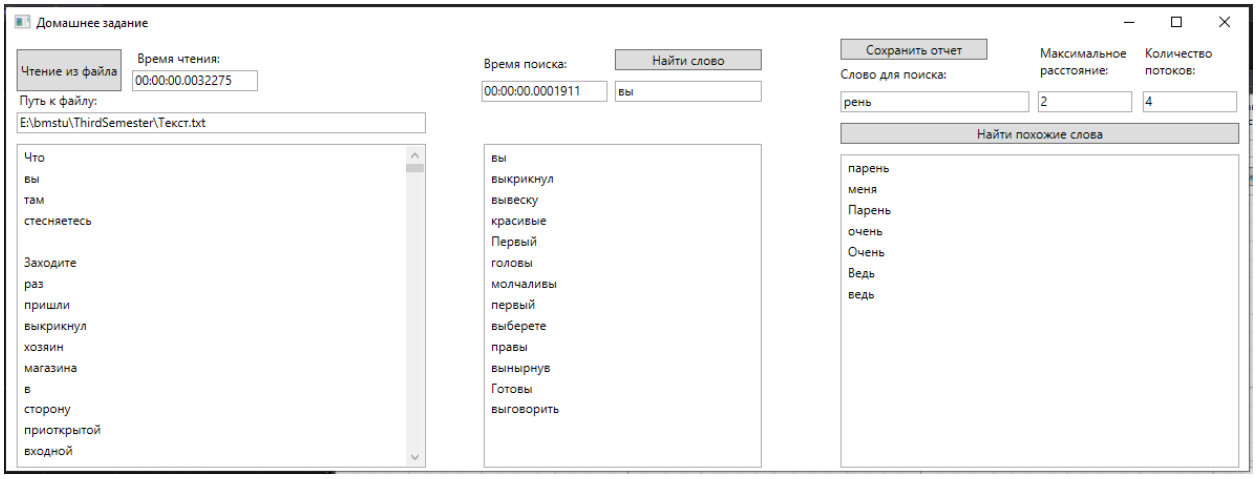
SubArrays.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Parallel
{
    /// <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;
        }
    }
}
```

Результат работы программы



И html-страничка (сохраненный отчет)

Отчет: E:\bmstu\ThirdSemester\Report_21_12_2020_091737.html

Время чтения из файла	00:00:00.0032275
Слово для поиска	рень
Максимальное расстояние для нечеткого поиска	2
Время четкого поиска	00:00:00.0001911
Результаты поиска	• парень • меня • Парень • очень • Очень • Ведь • ведь