

**Московский государственный технический университет им. Н.Э.  
Баумана**

Кафедра  
“Системы обработки информации и управления”  
(ИУ – 5)

**Отчет по домашнему заданию  
по дисциплине “Базовые компоненты Интернет-технологий”**

Выполнил:  
студент гр. ИУ5 - 31Б  
Аушева Лиза  
7 декабря 2018 г.

**Москва – 2018 г.**

## Домашнее задание

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

Form1

Класс

→ Form

Поля

buttonApprox

buttonExact

buttonExit

buttonLoadFile

buttonSaveReport

components

label1

label2

label3

label4

label5

label6

label7

label8

list

listBoxResult

textBoxApproxTime

textBoxExactTime

textBoxFileReadCount

textBoxFileReadTime

textBoxFind

textBoxMaxDist

textBoxThreadCount

textBoxThreadCountAll

Методы

ArrayThreadTask

buttonApprox\_Click

buttonExact\_Click

buttonExit\_Click

buttonLoadFile\_Click

buttonSaveReport\_Click

Dispose

Form1

InitializeComponent

SubArrays

Static Класс

Методы

DivideSubArrays

MinMax

Класс

Свойства

Max

Min

Методы

MinMax

EditDistance

Static Класс

Методы

Distance

ParallelSearchR...

Класс

Свойства

dist

ThreadNum

word

Program

Static Класс

Методы

Main

ParallelSearchT...

Класс

Свойства

maxDist

tempList

ThreadNum

wordPattern

## Текст программы

### Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace WindowsFormsFiles
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

### ParallelSearchThreadParam.cs

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

namespace WindowsFormsFiles
{
    /// <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; }
    }
}
```

## ParallelSearchResult.cs

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

namespace WindowsFormsFiles
{
    /// <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; }
    }
}
```

## MinMax.cs

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

namespace WindowsFormsFiles
{
    /// <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;

namespace WindowsFormsFiles
```

```

{
    /// <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;
        }
    }
}

```

## EditDistance.cs

```

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

namespace WindowsFormsFiles
{
    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
                int symbEqual = ((str1.Substring(i - 1, 1) == str2.Substring(j - 1,
m(s1[i],s2[j])
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];
    }
}

```

**Экранные формы с примерами выполнения программы:**

Поиск в файле

Чтение из файла

Время чтения из файла:

Количество уникальных слов в файле:

Слово для поиска:

Четкий поиск

Время четкого поиска:

Максимальное расстояние для нечеткого поиска:

Параллельный нечеткий поиск

Количество потоков:

10

Вычисленное количество потоков:

Время нечеткого поиска:

Сохранение отчета

Выход