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

Факультет ИУ

Кафедра ИУ5

Домашнее задание по курсу С#

Группа: ИУ5-32

Миронов С.В.

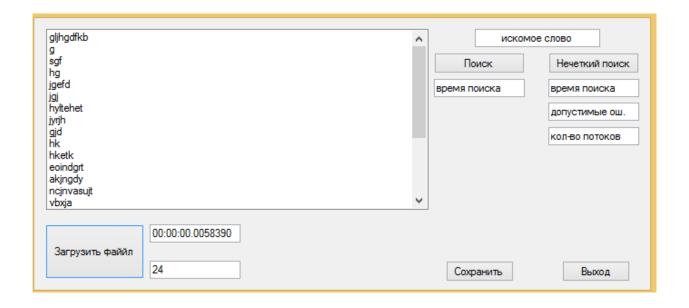
#### Домашнее задание

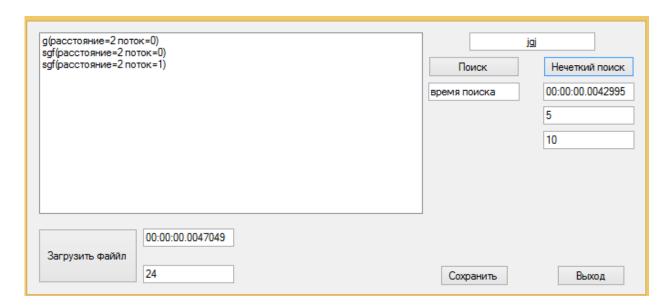
Пример реализации ДЗ рассмотрен в учебном пособии, глава «Пример многопоточного поиска в текстовом файле с использованием технологии Windows Forms».

Разработать программу, реализующую многопоточный поиск в файле.

- 1. Программа должна быть разработана в виде приложения Windows Forms на языке С#. 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.

### Результат выполнения программы:





## Код программы:

```
class MinMax
        public int Min { get; set; }
        public int Max { get; set; }
        public MinMax(int pmin, int pmax) { this.Min = pmin; this.Max = pmax; }
class ParallelSearchRes
    {
        public string word { get; set; }
        public int dist { get; set; }
        public int ThreadNum { get; set; }
class ParallelSearchThreadParam
   {
        public List<string> tempList { get; set; }
        public string wordPattern { get; set; }
        public int maxDist { get; set; }
        public int ThreadNum { get; set; }
    }
class SubArrays
       public static List<MinMax> DivideSubArrays(int beginIndex, int endIndex, int
subArraysCount)
        {
            List<MinMax> result = new List<MinMax>();
            if ((endIndex - beginIndex) <= subArraysCount)</pre>
            {
                result.Add(new MinMax(0, (endIndex - beginIndex)));
            }
            else
                int delta = (endIndex - beginIndex) / subArraysCount;
                int currentBegin = beginIndex;
                while ((endIndex - currentBegin) >= 2 * delta)
                    result.Add(new MinMax(currentBegin, currentBegin + delta));
                    currentBegin += delta;
                }
```

```
result.Add(new MinMax(currentBegin, endIndex));
}
return result;
}
```

• • •

#### Изменения в классе формы

```
private void buttonApprox_Click(object sender, EventArgs e)
            string word = this.textBoxFind.Text.Trim();
            if (!string.IsNullOrWhiteSpace(word) && listBoxResult.Items.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.textBoxThreads.Text.Trim(), out ThreadCount))
                    MessageBox.Show("Необходимо указать количество потоков"); return;
                }
                string wordUpper = word.ToUpper();
                List<Tuple<string, int>> tempList = new List<Tuple<string, int>>();
                Stopwatch t = new Stopwatch();
                t.Start();
                List<ParallelSearchRes> Result = new List<ParallelSearchRes>();
                List<MinMax> arrayDivList = SubArrays.DivideSubArrays(0,
listBoxResult.Items.Count, ThreadCount);
                int count = arrayDivList.Count;
                Task<List<ParallelSearchRes>>[] tasks = new
Task<List<ParallelSearchRes>>[count];
                for (int i = 0; i < count; i++)
                    List<string> tempTaskList = new List<string>();
         for (int i1=arrayDivList[i].Min;i1<=arrayDivList[i].Max - arrayDivList[i].Min; i1++ )</pre>
                        tempTaskList.Add(listBoxResult.Items[i1].ToString());
                    }
                    tasks[i] = new Task<List<ParallelSearchRes>>( Program.ArrayThreadTask, new
      ParallelSearchThreadParam() { tempList = tempTaskList, maxDist = maxDist, ThreadNum = i,
                                                                         wordPattern = word });
                    tasks[i].Start();
                Task.WaitAll(tasks);
                /*foreach (string str in listBoxResult.Items)
                    //Вычисление расстояния Дамерау-Левенштейна
                    int dist = ab5library.EditDistanse.Distance(str.ToUpper(), wordUpper);
```

```
//Если расстояние меньше порогового, то слово добавляется в результат
           if (dist <= maxDist)</pre>
           {
               tempList.Add(new Tuple<string, int>(str, dist));
           }
       }*/
       t.Stop();
       for (int i = 0; i < count; i++) { Result.AddRange(tasks[i].Result); }</pre>
       this.textBoxApproxTime.Text = t.Elapsed.ToString();
       this.listBoxResult.BeginUpdate();
       this.listBoxResult.Items.Clear();
       foreach (var x in Result)
       {
               string temp = x.word + "(расстояние=" + x.dist.ToString() + " поток=" +
                                                          x.ThreadNum.ToString() + ")";
           this.listBoxResult.Items.Add(temp);
       /*foreach (var x in tempList)
       {
           string temp = x.Item1 + "(расстояние=" + x.Item2.ToString() + ")";
           this.listBoxResult.Items.Add(temp);
       }*/
       this.listBoxResult.EndUpdate();
   }
   else
   {
       MessageBox.Show("Необходимо выбрать файл и ввести слово для поиска");
   }
}
```