Московский государственный технический университет имени Н.Э.Баумана (МГТУ им. Н.Э.Баумана)

ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ №4 и №5

«РАЗРАБОТКА ПРОГРАММЫ ДЛЯ РАБОТЫ С ФАЙЛАМИ»

и

«ВЫЧИСЛЕНИЕ РАССТОЯНИЯ ЛЕВЕНШТЕЙНА С ИСПОЛЬЩОВАНИЕМ АЛГОРИТМА ВАГНЕРА-ФИШЕРА»

ПО ДИСЦИПЛИНЕ «БАЗОВЫЕ КОМПОНЕНТЫ ИНТЕРНЕТ-ТЕХНОЛОГИЙ»

Выполнил(а): <u>Хапов А.В.</u> студент группы <u>ИУ5-31</u>

	Провер	ил: Гапанюк Ю.Е.
‹ ‹	>>	2017 г.

1. Задание

- №4. Разработать программу, реализующую работу с файлами.
- 1. Программа должна быть разработана в виде приложения Windows Forms на языке С#. По желанию вместо Windows Forms возможно использование WPF (Windows Presentation Foundation).
- 2. Добавить кнопку, реализующую функцию чтения текстового файла в список слов List.
- 3. Для выбора имени файла используется класс OpenFileDialog, который открывает диалоговое окно с выбором файла. Ограничить выбор только файлами с расширением «.txt».
- 4. Для чтения из файла рекомендуется использовать статический метод ReadAllText() класса File (пространство имен System.IO). Содержимое файла считывается методом ReadAllText() в виде одной строки, далее делится на слова с использованием метода Split() класса string. Слова сохраняются в список List.
- 5. При сохранении слов в список List дубликаты слов не записываются. Для проверки наличия слова в списке используется метод Contains().
- 6. Вычислить время загрузки и сохранения в список с использованием класса Stopwatch (пространство имен System.Diagnostics). Вычисленное время вывести на форму в поле ввода (TextBox) или надпись (Label).
- 7. Добавить на форму поле ввода для поиска слова и кнопку поиска. При нажатии на кнопку поиска осуществлять поиск введенного слова в списке. 10 Слово считается найденным, если оно входит в элемент списка как подстрока (метод Contains() класса string).
- 8. Добавить на форму список (ListBox). Найденные слова выводить в список с использованием метода «название_списка.Items.Add()». Вызовы метода «название_списка.Items.Add()» должны находится между вызовами методов «название_списка.BeginUpdate()» и «название_списка. EndUpdate()».
- 9. Вычислить время поиска с использованием класса Stopwatch. Вычисленное время вывести на форму в поле ввода (TextBox) или надпись (Label).
- №5. Разработать программу, реализующую вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера.
- 1. Программа должна быть разработана в виде библиотеки классов на языке С#.
- 2. Использовать самый простой вариант алгоритма без оптимизации. 3. Дополнительно возможно реализовать вычисление расстояния Дамерау-Левенштейна (с учетом перестановок соседних символов).
- 4. Модифицировать предыдущую лабораторную работу, вместо поиска подстроки используется вычисление расстояния Левенштейна.
- 5. Предусмотреть отдельное поле ввода для максимального расстояния. Если расстояние Левенштейна между двумя строками больше максимального, то строки считаются несовпадающими и не выводятся в список результатов.

2. Текст программы

public static class EditDistance

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Ling;
using System. Text;
using System.Windows.Forms;
using System.IO;
using System. Diagnostics;
namespace lab4
  public partial class Form1 : Form
    public Form1()
       InitializeComponent();
    /// <summary>
    /// Список слов
    /// </summary>
    List<string> list = new List<string>();
    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");
    }
```

```
/// <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 \le str1Len; i++) matrix[i, 0] = i;
         for (int j = 0; j \le str2Len; j++) matrix[0, j] = j;
         //Вычисление расстояния Дамерау-Левенштейна
         for (int i = 1; i \le str1Len; i++)
         {
            for (int j = 1; j \le 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) \&\& (i > 1) \&\&
                 (str1.Substring(i - 1, 1) == str2.Substring(i - 2, 1)) &&
                 (str1.Substring(i - 2, 1) == str2.Substring(i - 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];
       }
    }
```

```
private void buttonExit Click 1(object sender, EventArgs e)
  this.Close();
  //Application.Exit();
}
private void buttonLoadFile Click 1(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");
}
private void buttonExact Click 1(object sender, EventArgs e)
  //Слово для поиска
  string word = this.textBoxFind.Text.Trim();
  //Если слово для поиска не пусто
  if (!string.lsNullOrWhiteSpace(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.textBoxExactTime.Text = t.Elapsed.ToString();
         this.listBoxResult.BeginUpdate();
         //Очистка списка
         this.listBoxResult.ltems.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 buttonApprox_Click_1(object sender, EventArgs e)
       //Слово для поиска
       string word = this.textBoxFind.Text.Trim();
       //Если слово для поиска не пусто
       if (!string.lsNullOrWhiteSpace(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;
         }
         //Слово для поиска в верхнем регистре
         string wordUpper = word.ToUpper();
         //Временные результаты поиска
         List<Tuple<string, int>> tempList = new List<Tuple<string, int>>();
         Stopwatch t = new Stopwatch();
         t.Start();
         foreach (string str in list)
            //Вычисление расстояния Дамерау-Левенштейна
            int dist = EditDistance.Distance(str.ToUpper(), wordUpper);
            //Если расстояние меньше порогового, то слово добавляется в результат
```

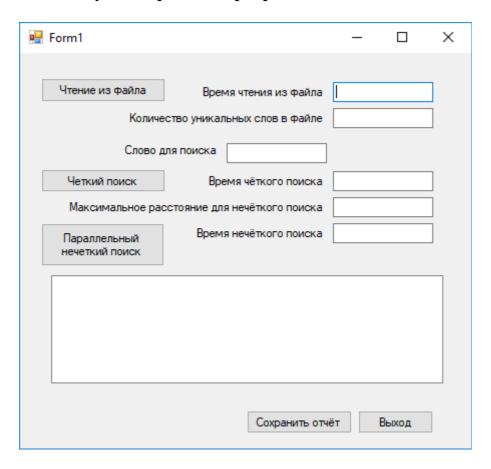
```
if (dist <= maxDist)
            {
              tempList.Add(new Tuple<string, int>(str, dist));
           }
         }
         t.Stop();
         this.textBoxApproxTime.Text = t.Elapsed.ToString();
         this.listBoxResult.BeginUpdate();
         //Очистка списка
         this.listBoxResult.Items.Clear();
         //Вывод результатов поиска
         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");
       }
    }
    private void buttonSaveReport_Click_1(object sender, EventArgs e)
       //Имя файла отчета
       string TempReportFileName = "Report " +
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>");
```

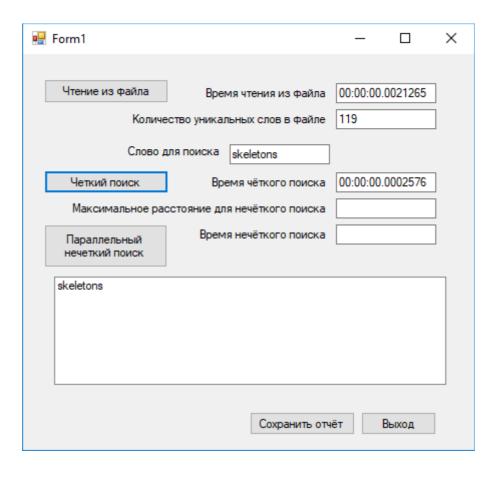
```
b.AppendLine("");
   b.AppendLine("");
    b.AppendLine("Время чтения из файла");
    b.AppendLine("" + this.textBoxFileReadTime.Text + "");
   b.AppendLine("");
    b.AppendLine("");
    b.AppendLine("Количество уникальных слов в файле);
    b.AppendLine("" + this.textBoxFileReadCount.Text + "");
    b.AppendLine("");
    b.AppendLine("");
    b.AppendLine("Слово для поиска");
    b.AppendLine("" + this.textBoxFind.Text + "");
    b.AppendLine("");
   b.AppendLine("");
   b.AppendLine("Maксимальное расстояние для нечеткого поиска
    b.AppendLine("" + this.textBoxMaxDist.Text + "");
    b.AppendLine("");
    b.AppendLine("");
    b.AppendLine("Время четкого поиска");
    b.AppendLine("" + this.textBoxExactTime.Text + "");
   b.AppendLine("");
   b.AppendLine("");
    b.AppendLine("Время нечеткого поиска");
    b.AppendLine("" + this.textBoxApproxTime.Text + "");
    b.AppendLine("");
    b.AppendLine("");
    b.AppendLine("Peзультаты поиска");
    b.AppendLine(">"); b.AppendLine("");
   foreach (var x in this.listBoxResult.ltems)
      b.AppendLine("" + x.ToString() + "");
   b.AppendLine("");
    b.AppendLine("");
   b.AppendLine("");
    b.AppendLine("");
    b.AppendLine("</body>");
    b.AppendLine("</html>");
   //Сохранение файла
    File.AppendAllText(ReportFileName, b.ToString());
    MessageBox.Show("Отчет сформирован. Файл: " + ReportFileName);
private void listBoxResult SelectedIndexChanged(object sender, EventArgs e)
```

} }

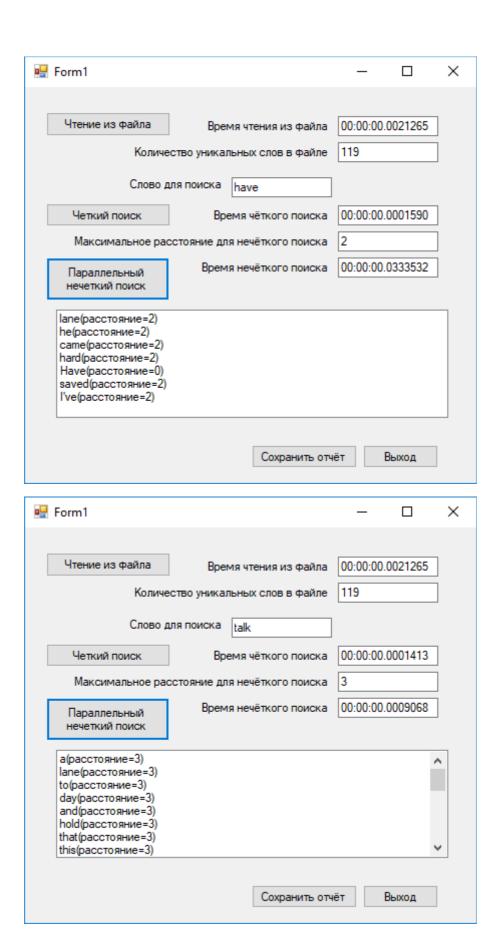
```
}
```

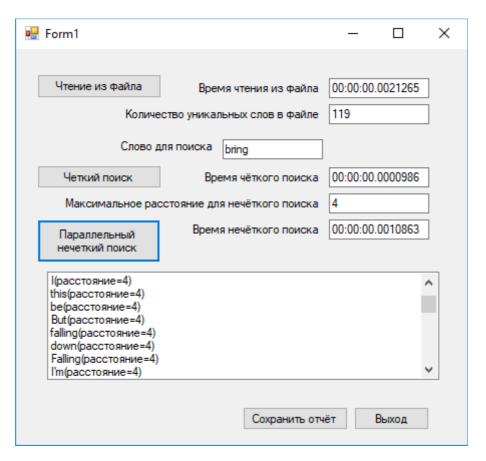
3. Результат работы программы





⊞ Form1	_		×		
Чтение из файла Время чтения из файла Количество уникальных слов в файле	00:00:0	0.0021265			
Слово для поиска talk					
Четкий поиск Время чёткого поиска	00:00:0	0.0000986			
Максимальное расстояние для нечёткого поиска	1				
Параллельный Время нечёткого поиска нечеткий поиск	00:00:0	0.0034636			
(Talk(расстояние=1) Talk(расстояние=0)					
Сохранить отчёт Выход					





4. Диаграмма классов

