**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИН. Э.БАУМАНА**

Лабораторная работа №5 по курсу

*Базовые компоненты Интернет технологий*

ПАРАЛЛЕЛЬНЫЙ ПОИСК В WINDOWSFORMS

студента 2-го курса

РТ факультета группы РТ5-31

Калин Владимир

Москва – 2016г.

**Цель работы:**   
Разработать программу для работы с файлами при алгоритма вычисления расстояния Левенштейна

**Описание программы:**  
Программа разработана в виде библиотеки классов на языке C#. Использован самый простой вариант алгоритма без оптимизации. Модифицирована предыдущая лабораторная работа, вместо поиска подстроки используется вычисление расстояния Левенштейна. Предусмотрено отдельное поле ввода для максимального расстояния. Если расстояние Левенштейна между двумя строками больше максимального, то строки считаются несовпадающими и не выводятся в список результатов.

**Текстпрограммы**

using System;

usingSystem.Collections.Generic;

usingSystem.ComponentModel;

usingSystem.Data;

usingSystem.Drawing;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Windows.Forms;

using System.IO;

usingSystem.Diagnostics;

usingSystem.Threading.Tasks;

namespaceWindowsFormsFiles

{

publicpartialclassForm1 :Form

{

public Form1()

{

InitializeComponent();

}

///<summary>

///Списокслов

///</summary>

List<string> list = newList<string>();

privatevoidbuttonLoadFile\_Click(object sender, EventArgs e)

{

OpenFileDialogfd = newOpenFileDialog();

fd.Filter = "текстовыефайлы|\*.txt";

if (fd.ShowDialog() == DialogResult.OK)

{

Stopwatch t = newStopwatch();

t.Start();

//Чтениефайлаввидестроки

string text = File.ReadAllText(fd.FileName);

//Разделительные символы для чтения из файла

char[] separators = newchar[] {' ','.',',','!','?','/','\t','\n'};

string[] textArray = text.Split(separators);

foreach (stringstrTempintextArray)

{

//Удаление пробелов в начале и конце строки

stringstr = 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("Можетвсё-такивыберетефайл?");

}

}

privatevoidbuttonExact\_Click(object sender, EventArgs e) //Кнопкачёткогопоиска

{

//Словодляпоиска

string word = this.textBoxFind.Text.Trim();

//Если слово для поиска не пусто

if(!string.IsNullOrWhiteSpace(word) &&list.Count> 0)

{

//Слово для поиска в верхнем регистре

stringwordUpper = word.ToUpper();

//Временные результаты поиска

List<string>tempList = newList<string>();

Stopwatch t = newStopwatch();

t.Start();

foreach (stringstrin list)

{

if (str.ToUpper().Contains(wordUpper))

{

tempList.Add(str);

}

}

t.Stop();

this.textBoxExactTime.Text = t.Elapsed.ToString();

this.listBoxResult.BeginUpdate();

//Очисткасписка

this.listBoxResult.Items.Clear();

//Выводрезультатовпоиска

foreach (stringstrintempList)

{

this.listBoxResult.Items.Add(str);

}

this.listBoxResult.EndUpdate();

}

else

{

MessageBox.Show("ЛучшебыВывыбралифайливвелисловодляпоиска");

}

}

privatevoidbuttonApprox\_Click(object sender, EventArgs e) //Кнопкапараллельногопоиска

{

//Словодляпоиска

string word = this.textBoxFind.Text.Trim();

//Если слово для поиска не пусто

if(!string.IsNullOrWhiteSpace(word) &&list.Count> 0)

{

intmaxDist;

if(!int.TryParse(this.textBoxMaxDist.Text.Trim(), outmaxDist))

{

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;

}

intThreadCount;

if(!int.TryParse(this.textBoxThreadCount.Text.Trim(), outThreadCount))

{

MessageBox.Show("Необходимо указать количество потоков");

return;

}

Stopwatch timer = newStopwatch();

timer.Start(); //Начало параллельного поиска

//Результирующий список

List<ParallelSearchResult> Result = newList<ParallelSearchResult>();

//Деление списка на фрагменты для параллельного запуска в потоках

List<MinMax>arrayDivList = SubArrays.DivideSubArrays(0, list.Count, ThreadCount);

intcount = arrayDivList.Count;

//Количество потоков соответствует количеству фрагментов массива

Task<List<ParallelSearchResult>>[] tasks = newTask<List<ParallelSearchResult>>[count];

//Запускпотоков

for (inti = 0; i< count; i++)

{

//Создание временного списка, чтобы потоки не работали параллельно с одной коллекцией

List<string>tempTaskList = list.GetRange(arrayDivList[i].Min, arrayDivList[i].Max - arrayDivList[i].Min);

tasks[i] = newTask<List<ParallelSearchResult>>(

//Метод, который будет выполняться в потоке

ArrayThreadTask,

//Параметрыпотока

newParallelSearchParam()

{

tempList = tempTaskList,

maxDist = maxDist,

ThreadNum = i,

wordPattern = word

});

//Запускпотока

tasks[i].Start();

}

Task.WaitAll(tasks);

timer.Stop();

//Объединениерезультатов

for (inti = 0; i< count; i++)

{

Result.AddRange(tasks[i].Result);

}

timer.Stop(); //Конец параллельного поиска

//Вывод результатов

//Время поиска

this.textBoxApproxTime.Text = timer.Elapsed.ToString();

//Вычисленное количество потоков

this.textBoxThreadCountAll.Text = count.ToString();

//Начало обновления списка результатов

this.listBoxResult.BeginUpdate();

//Очистка списка

this.listBoxResult.Items.Clear();

//Вывод результатов поиска

foreach (var x inResult)

{

stringtemp = x.word + "(расстояние=" + x.dist.ToString() + " поток=" + x.ThreadNum.ToString() + ")";

this.listBoxResult.Items.Add(temp);

}

//Окончание обновления списка результатов

this.listBoxResult.EndUpdate();

}

else

{

MessageBox.Show("Лучше бы Вы выбрали файл и ввели слово для поиска");

}

}

///<summary>

/// Выполняется в параллельном потоке для поиска строк

///</summary>

publicstaticList<ParallelSearchResult>ArrayThreadTask(objectparamObj)

{

ParallelSearchParamparam = (ParallelSearchParam)paramObj;

//Словодляпоискавверхнемрегистре

stringwordUpper = param.wordPattern.Trim().ToUpper();

//Результатыпоискаводномпотоке

List<ParallelSearchResult> Result = newList<ParallelSearchResult>();

//Перебор всех слов во временном списке данного потока

foreach (stringstrinparam.tempList)

{

//ВычислениерасстоянияДамерау-Левенштейна

intdist = EditDistance.Distance(str.ToUpper(), wordUpper);

//Если расстояние меньше порогового, то слово добавляется в результат

if (dist<= param.maxDist)

{

ParallelSearchResult temp = newParallelSearchResult()

{

word = str,

dist = dist,

ThreadNum = param.ThreadNum

};

Result.Add(temp);

}

}

return Result;

}

privatevoidbuttonExit\_Click(object sender, EventArgs e)

{

this.Close();

}

privatevoid Form1\_Load(object sender, EventArgs e)

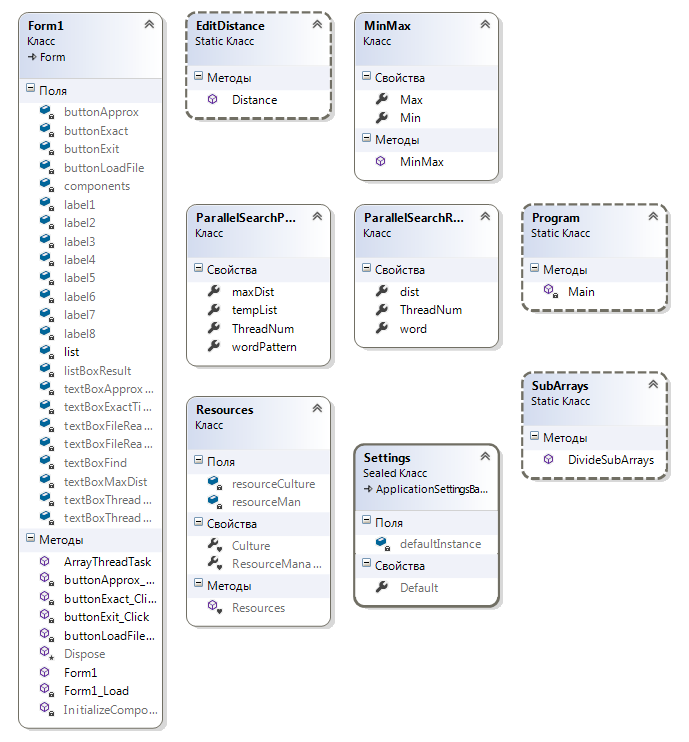
{

}

}

}

**Дерево классов**



**Результат работы программы**

