Министерство образования республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

**Институт информационных технологий**

Специальность ПОИТ

**КОНТРОЛЬНАЯ РАБОТА**

По дисциплине: «Основы ERP-систем»

Студент-заочник\_4\_ курса

Группы №\_\_581072\_\_\_\_\_\_

ФИО Вежновец \_\_\_\_\_\_\_\_

Виталина Валерьевна \_\_\_

Адрес\_Таллин, Пae 57-74

Тел. \_+ 37256142729\_\_\_\_\_

Проверила: Коренская Ирина Николаевна

Минск, 2018

**Задание 1**. «Применение законов Зипфа для обработки текстовой информации»

**Цель работы**: ознакомиться с законами Зипфа и научиться их применять для определения заданных характеристик текстовой информации.

**Порядок выполнения работы**:

1. Изучить теоретическую часть работы.
2. Реализовать алгоритм проверки первого закона Зипфа.
3. Реализовать алгоритм проверки второго закона Зипфа.
4. Оформить отчет по контрольной работе.

**Исходные данные**:

Тексты размером 1-2 страницы на русском, белорусском, английском языках.

**Выходные данные**:

Вычисленные характеристики Зипфа и наборы ключевых слов для исходных текстов.

**Выполнение работы:**

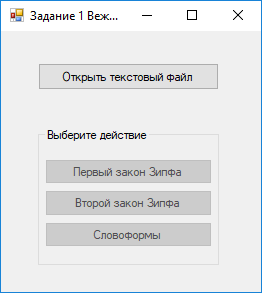
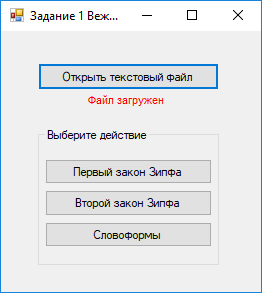
 

Рисунок 1 – Главная форма программы

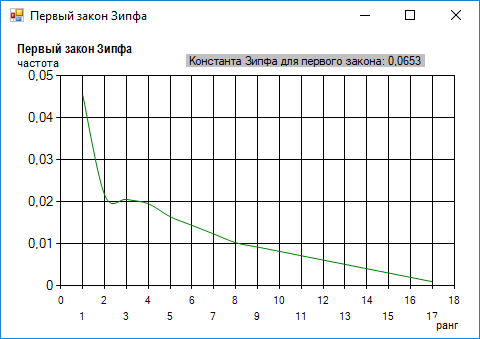


Рисунок 2 – Форма «Первый закон Зипфа»

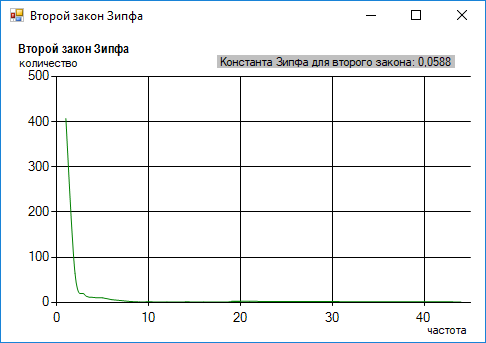


Рисунок 3 – Форма «Второй закон Зипфа»

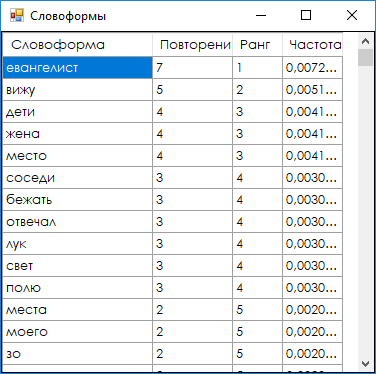


Рисунок 4 – Форма «Словоформы»

**Код программы:**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Threading.Tasks;

namespace l1

{

public partial class Form2 : Form

{List<string> wordsq = new List<string>();

public double zipfLaw1;

public double zipfLaw2;

public double temp;

public double extra;

public Form2()

{

InitializeComponent();

var col1 = new DataGridViewColumn

{

HeaderText = "Словоформа",

Name = "word",

ReadOnly = true,

Resizable = DataGridViewTriState.True,

Width = 150,

CellTemplate = new DataGridViewTextBoxCell()

};

var col2 = new DataGridViewColumn

{

HeaderText = "Повторений",

Name = "repeats",

ReadOnly = true,

Resizable = DataGridViewTriState.True,

Width = 50,

CellTemplate = new DataGridViewTextBoxCell()

};

var col3 = new DataGridViewColumn

{

HeaderText = "Ранг",

Name = "rank",

ReadOnly = true,

Resizable = DataGridViewTriState.True,

Width = 50,

CellTemplate = new DataGridViewTextBoxCell()

};

var col4 = new DataGridViewColumn

{

HeaderText = "Частота",

Name = "freq",

ReadOnly = true,

Resizable = DataGridViewTriState.True,

Width = 50,

CellTemplate = new DataGridViewTextBoxCell()

};

dataGridView1.Columns.Add(col1);

dataGridView1.Columns.Add(col2);

dataGridView1.Columns.Add(col3);

dataGridView1.Columns.Add(col4);

}

private async void button1\_Click(object sender, EventArgs e)

{

label1.Visible = false;

var words = new Dictionary<string, int>();

var ranks = new List<int>();

var numRanks = new SortedDictionary<int, int>();

var ofd = new OpenFileDialog();

var numWords = 0;

ofd.Filter = "Text files(\*.txt)|\*.txt";

stop\_l("D:\\Сессия\\Основы ERP-систем\\texts\\text!.txt");

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

{

button2.Enabled = false;

button5.Enabled = false;

button4.Enabled = false;

double ZipfC1 = 0;

double ZipfC2 = 0;

await Task.Run(() =>

{ var sr = new StreamReader(ofd.FileName);

while (!sr.EndOfStream)

{

var s = sr.ReadLine();

var w = s.Split();

foreach (var str in w)

{

var tmp = str.Trim('\'', '\"', '.', ',', ';', ':',

'?', '!', '-', '\_', '-', '+', '=', '/', '\*', '–',

'(', ')', '1', '2', '3', '4', '5', '6', '7',

'8', '9', '0', '—');

tmp = tmp.ToLowerInvariant();

if (tmp != "")

{

numWords++;

if (words.ContainsKey(tmp))

{

words[tmp]++;

}

else

{

words.Add(tmp, 1);

}

}

}

}

sr.Close();

foreach (var k in words)

{

if (!ranks.Contains(k.Value))

{

ranks.Add(k.Value);

}

if (numRanks.ContainsKey(k.Value))

{

numRanks[k.Value]++;

}

else

{

numRanks.Add(k.Value, 1);

}

}

ranks.Sort();

ranks.Reverse();

var wordsList = words.ToList();

wordsList.Sort(

delegate (KeyValuePair<string, int> pair1,

KeyValuePair<string, int> pair2)

{

return pair1.Value.CompareTo(pair2.Value);

});

wordsList.Reverse();

var sw = new StreamWriter(ofd.FileName + "\_keywords.log", false, Encoding.UTF8);

var \_beg = ranks[(int)(0 \* (double)ranks.Count)];

var \_end = ranks[(int)(1 \* (double)ranks.Count) - 1];

if (checkBox1.Checked)

{

sw.WriteLine("Количество слов:" + words.Count.ToString());

}

var lastrang = 0;

var lastRep = -122;

dataGridView1.Rows.Clear();

foreach (var k in wordsList)

{

if ((k.Value <= \_beg) && (k.Value >= \_end) && (!wordsq.Contains(k.Key)))

{

if (checkBox1.Checked && lastrang < 2000)

{

var rank = k.Value == lastRep ? lastrang : ++lastrang;

lastRep = k.Value;

dataGridView1.Rows.Add(k.Key, k.Value, rank, (double)k.Value / (double)numWords);

}

else

{

sw.WriteLine(k.Key);

}

}

}

sw.Close();

var DataSer1 = new System.Windows.Forms.DataVisualization.Charting.Series

{

ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline,

Color = Color.Green

};

chart1.Series.Clear();

var DataSer2 = new System.Windows.Forms.DataVisualization.Charting.Series

{

ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline,

Color = Color.Green

};

chart2.Series.Clear();

for (var i = 0; i <= ranks.Count - 1; i++)

{

ZipfC1 += (ranks[i] \* (double)(i + 1) / numWords);

DataSer1.Points.AddXY(i + 1, (double)ranks[i] / numWords);

}

foreach (var kvp in numRanks)

{

if (kvp.Key <= 100)

{

DataSer2.Points.AddXY(kvp.Key, kvp.Value);

ZipfC2 += kvp.Key \* (double)kvp.Value / numWords;

}

}

ZipfC1 /= ranks.Count;

ZipfC2 /= numRanks.Count;

chart1.ResetAutoValues();

chart1.Series.Add(DataSer1);

chart2.ResetAutoValues();

chart2.Series.Add(DataSer2);

GraphHelper.DataSer1 = DataSer1;

GraphHelper.GridRows = dataGridView1.Rows;

GraphHelper.DataSer2 = DataSer2;

GraphHelper.ZipfC1 = ZipfC1.ToString("f4");

GraphHelper.ZipfC2 = ZipfC2.ToString("f4");

});

label1.Visible = true;

button2.Enabled = true;

button5.Enabled = true;

button4.Enabled = true;

}

}

public void stop\_l(string str\_n)

{

var sr = new StreamReader(str\_n);

while (!sr.EndOfStream)

{

var s = sr.ReadLine();

var w = s.Split();

foreach (var str in w)

{

var tmp = str.Trim('\'');

tmp = tmp.ToLowerInvariant();

if (tmp != "")

{

wordsq.Add(tmp);

}

}

}

sr.Close();

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

switch (comboBox1.SelectedIndex)

{

case 0:

chart1.Visible = true;

chart2.Visible = false;

dataGridView1.Visible = false;

break;

case 1:

chart1.Visible = false;

chart2.Visible = true;

dataGridView1.Visible = false;

break;

default:

chart1.Visible = false;

chart2.Visible = false;

dataGridView1.Visible = true;

break;

}

}

private void button2\_Click(object sender, EventArgs e)

{

new Graph1().ShowDialog();

}

private void button3\_Click(object sender, EventArgs e)

{

new GridForm().ShowDialog();

}

private void button4\_Click(object sender, EventArgs e)

{

new Graph2().ShowDialog();

}

**Вывод:** при выполнении задания контрольной работы был реализован алгоритм проверки первого закона Зипфа, реализован алгоритм проверки второго закона Зипфа.

Для каждого текста было определено количество вхождений слов и их ранг, а также была вычислена константа. Константа зависит от вероятности обнаружения слова в тексте на ранг частоты.

Значение константы *С*в разных языках различно, но внутри одной языковой группы остается неизменным, в независимости от текста. Так, например, для английских текстов константа Зипфа равна приблизительно 0,1.

Согласно второго закона Зипфа частота и количество слов, входящих в текст с одинаковой частотой связаны между собой. Согласно получившегося графика наиболее значимые слова лежат в средней части диаграммы. Получившаяся кривая будет сохранять свои параметры для всех текстов в пределах одного языка.

**Задание 2.** «Поиск текстовой информации по заданному набору ключевых слов»

**Цель работы:**реализовать поиск текстовой информации по документу-образцу.

**Порядок выполнения работы:**

1. Ознакомиться с теоретической частью работы.
2. Реализовать алгоритм поиска текстовой информации по документу-образцу.
3. Оформить отчет.

**Исходные данные**: коллекция документов из 10-12 текстов на русском и английском языках. Результат первой лабораторной работы: вектор ключевых слов, построенный по документу-образцу.

**Выходные данные:** документы, найденные в результате работы поисковой машины и соответствующие запросу поиска.

**Выполнение работы:**

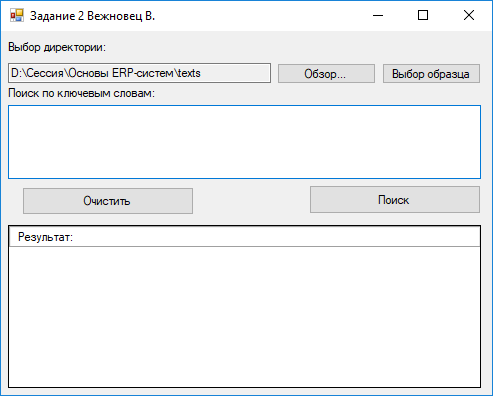


Рисунок 5 – Главная форма программы

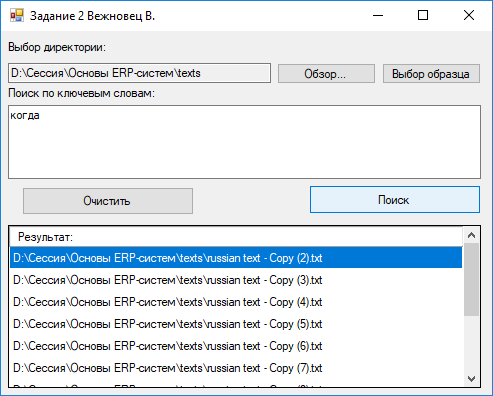


Рисунок 6 – Результаты поиска по ключевому слову

**Код программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

namespace l2

{

public partial class MainForm : Form

{

private string currentFolder = "D:\\Сессия\\Основы ERP-систем\\texts";

private List<string> wordsForSearch;

private List<string> wordsq = new List<string>();

public MainForm()

{

InitializeComponent();

wordsForSearch = new List<string>();

textBox2.Text = currentFolder;

}

private void button2\_Click(object sender, EventArgs e)

{

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

{

currentFolder = folderBrowserDialog1.SelectedPath;

textBox2.Text = currentFolder;

}

}

private void button1\_Click(object sender, EventArgs e)

{

wordsForSearch.Clear();

dataGridView1.Rows.Clear();

var src = textBox1.Text;

var w = src.Split();

foreach (var str in w)

{

var tmp = str.Trim('\'', '\"', '.', ',', ';', ':', '?', '!', '-', '\_', '(', ')', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0');

tmp = tmp.ToLowerInvariant();

if ((tmp != "")&&(!wordsForSearch.Contains(tmp)))

{

wordsForSearch.Add(tmp);

}

}

var di = new DirectoryInfo(currentFolder);

var files = di.GetFiles("\*.txt");

for (var i = 0; i <= files.Length - 1; i++)

{

var sg = new StatGenerator(files[i].FullName);

var dic = sg.GetStatistics();

var chk = false;

foreach (var str in wordsForSearch)

{

if (dic.ContainsKey(str))

{

chk = true;

}

}

if (chk)

{

dataGridView1.Rows.Add(new DataGridViewRow());

dataGridView1.Rows[dataGridView1.Rows.Count-1].Cells[0].Value = files[i].FullName;

}

}

}

private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

var p = new System.Diagnostics.Process

{

StartInfo = {FileName = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString()}

};

p.Start();

}

private void button3\_Click(object sender, EventArgs e)

{

var words = new Dictionary<string, int>();

var ranks = new List<int>();

var numRanks = new SortedDictionary<int, int>();

var ofd = new OpenFileDialog();

var numWords = 0;

ofd.Filter = "Text files(\*.txt)|\*.txt";

stop\_l("D:\\Сессия\\Основы ERP-систем\\texts\\text!.txt");

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

{

var sr = new StreamReader(ofd.FileName);

while (!sr.EndOfStream)

{

var s = sr.ReadLine();

var w = s.Split();

foreach (var str in w)

{

var tmp = str.Trim('\'', '\"', '.', ',', ';', ':',

'?', '!', '-', '\_',

'(', ')', '1', '2', '3', '4', '5', '6', '7',

'8', '9', '0');

tmp = tmp.ToLowerInvariant();

if (tmp != "")

{

numWords++;

if (words.ContainsKey(tmp))

{

words[tmp]++;

}

else

{

words.Add(tmp, 1);

}

}

}

}

sr.Close();

foreach (var k in words)

{

if (!ranks.Contains(k.Value))

{

ranks.Add(k.Value);

}

if (numRanks.ContainsKey(k.Value))

{

numRanks[k.Value]++;

}

else

{

numRanks.Add(k.Value, 1);

}

}

ranks.Sort();

ranks.Reverse();

var wordsList = words.ToList();

wordsList.Sort(

delegate(KeyValuePair<string, int> pair1,

KeyValuePair<string, int> pair2)

{

return pair1.Value.CompareTo(pair2.Value);

});

wordsList.Reverse();

var sw = new StreamWriter(ofd.FileName + "\_keywords.log", false, Encoding.UTF8);

var \_beg = ranks[(int)(0.3 \* (double)ranks.Count)];

var \_end = ranks[(int)(0.5 \* (double)ranks.Count)];

foreach (var k in wordsList)

{

if ((k.Value <= \_beg) && (k.Value >= \_end) && (k.Key.Length > 2) && (!wordsq.Contains(k.Key)))

{

textBox1.Text += k.Key + " ";

}

}

sw.Close();

double ZipfC1 = 0;

double ZipfC2 = 0;

for (var i = 0; i <= ranks.Count - 1; i++)

{

ZipfC1 += (ranks[i] \* (double)(i + 1) / numWords);

}

foreach (var kvp in numRanks)

{

if (kvp.Key <= 100)

{

ZipfC2 += kvp.Key \* (double)kvp.Value / numWords;

}

}

ZipfC1 /= ranks.Count;

ZipfC2 /= numRanks.Count;

}

}

public void stop\_l(string str\_n)

{

var sr = new StreamReader(str\_n);

while (!sr.EndOfStream)

{

var s = sr.ReadLine();

var w = s.Split();

foreach (var str in w)

{

var tmp = str.Trim('\'');

tmp = tmp.ToLowerInvariant();

if (tmp != "")

{

wordsq.Add(tmp);

}

}

}

sr.Close();

}

private void button4\_Click(object sender, EventArgs e)

{

textBox1.Clear();

}

}

}

**Вывод:** при выполнении задания контрольной работы был реализован алгоритм поиска текстовой информации по документу. Была вычислена частота вхождения каждого термина. Затем термины были отсортированы в порядке убывания их частоты вхождения. Далее выбран диапазон частот из середины упорядоченного списка, составлен запрос из отобранных слов в порядке их следования в списке терминов. Далее запрос обрабатывается поисковой системой.

При выполнении задания использовался тематический поиск или поиск «по тексту». Общая схема такого поиска заключается в формулировании некоторого запроса пользователем относительно содержания документа и отборе множества доступных документов тех, которые удовлетворяют запросу.