**Министерство образования и науки Российской Федерации**

федеральное государственное автономное образовательное учреждение

высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт Кибернетики

Направление подготовки Информационные системы и технологии

Кафедра Автоматики и компьютерных систем

**ОТЧЕТ  
по лабораторной работе №2**

Алгоритмы обработки текстовых данных

по дисциплине Технологии обработки информации

Выполнила студентка группы 8И4А \_\_\_\_\_\_\_\_\_\_ Леонова А. П.

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ 2017 г.

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_ Полищук В.Ю.

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ 2017 г.

Томск 2017 г.

**Задание 1:** Построить облако тегов по тексту.

Требования:

\* Для построения тегов использовать только имена существительные в именительном падеже (если не получится – любые слова, но в своей начальной форме)

\* Длина слов, подлежащих анализу – не менее 3 символов(можно и 4)

\* Указывать статистику вхождения слова в текст (и всех его форм)

**Расстояние Левенштейна:**

public static int LevenshteinDistance(string string1, string string2)

{

int m = string1.Length;

int n = string2.Length;

int count;

int[,] d = new int[m + 1, n + 1];

for (int i = 0; i <= m; i++)

d[i, 0] = i;

for (int j = 0; j <= n; j++)

d[0, j] = j;

for (int i = 1; i <= m; i++)

{

for (int j = 1; j <= n; j++)

{

if (string2[j - 1] == string1[i - 1])

count = 0;

else

count = 1;

d[i, j] = Math.Min(Math.Min(d[i - 1, j] + 1, d[i, j - 1] + 1), d[i - 1, j - 1] + count);

}

}

return d[m, n];

}

**Построение облака:**

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

{

int num = 0;

foreach (var word in words\_text.Where(w => w.Length > 4 && words.Any(word => Canonize.LevenshteinDistance(word, w) < 2)))

{

var count = words\_text.Count(w => w == word);

if (count > 0)

{

text.Add(word);

mas[num] = count;

num++;

}

}

WriteCloud();

}

**Результат:**

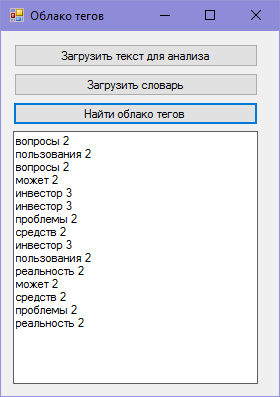


Рисунок 1 – Результат работы программы

**Задание 2:** Реализовать алгоритм вычисления разницы между двумя текстами (антиплагиат)

Требования:

\* Для сравнения используется 10 исходных текстов размером от 500 до 1000 слов.

\* Показать разницу между текстами. Показать общее число совпадений, процентное соотношение совпадений и все оформить в виде инфографики. (Опциональное пожелание – вывести наиболее подходящий текст и все совпадения подсветить зеленым цветом)

\* Время для анализа по всем текстам – не более 10 минут. (Можем опять соревнование на самую быструю реализацию провести)

**Канонизация текста:**

public string Canonize(string Data)

{

words = "";

string[] split = Data.Split( new Char[] { ' ', '.', ',', '!', '?', ':', ';', '(', ')', '-', '\n' });

foreach (string s in split)

{

if (s.Trim() != "")

words += s.ToLower() + " ";

words = words.Replace(" это ", " как ").Replace(" как ", " так ").Replace(" так ", " и ").

Replace(" и ", " в ").Replace(" в ", " над ").Replace(" над ", " к ").Replace(" к ", " до ").

Replace(" до ", " не ").Replace(" не ", " на ").Replace(" на ", " но ").Replace(" но ", " за ").

Replace(" за ", " то ").Replace(" то ", " с ").Replace(" с ", " ли ").Replace(" ли ", " а ").

Replace(" а ", " во ").Replace(" во ", " от ").Replace(" от ", " со ").Replace(" со ", " для ").Replace(" для ", " о ").

Replace(" о ", " же ").Replace(" же ", " ну ").Replace(" ну ", " вы ").Replace(" вы ", " бы ").Replace(" бы ", " что ").

Replace(" что ", " кто ").Replace(" кто ", " он ").Replace(" он ", " она ").Replace(" она ", " ");

}

return words;

}

**Алгоритм Шинглов:**

public StringBuilder Shingle(string Data)

{

Canonize(Data);

sBuilder = new StringBuilder();

sBuilder1 = new StringBuilder();

string shingle = "";

string hash = "";

int len = 5;

string[] split = words.Split(new Char[] { ' ' });

for (int i = 0; i < split.Length - len; i++)

{

if (i > 0)

shingle = "\n";

for (int j = i; j < i + len; j++)

{

shingle = shingle + split[j] + " ";

}

sBuilder.Append(shingle);

}

string[] splitagain = sBuilder.ToString().Split(new Char[] { '\n' });

foreach (string sa in splitagain)

{

using (MD5 md5Hash = MD5.Create())

{

hash = "\n";

hash += MD5hash(md5Hash, sa);

}

sBuilder1.Append(hash);

}

return sBuilder1;

}

**Кодирование в хэши:**

static string MD5hash (MD5 md5Hash, string input)

{

byte[] data = md5Hash.ComputeHash(Encoding.UTF8.GetBytes(input));

StringBuilder sBuilder = new StringBuilder();

for (int i = 0; i < data.Length; i++)

{

sBuilder.Append(data[i].ToString("x2"));

}

return sBuilder.ToString();

}

**Результат:**

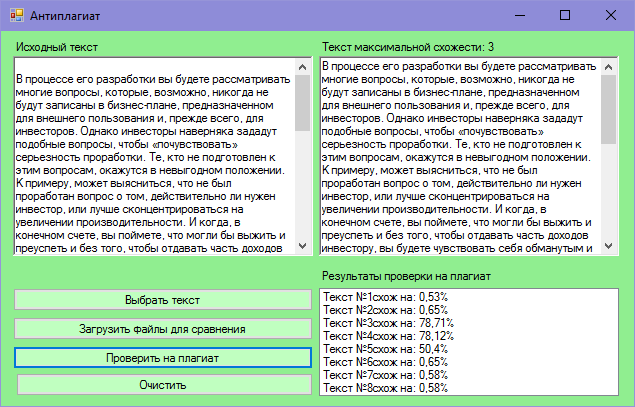


Рисунок 2 – Результат работы программы

**Вывод:**

В ходе лабораторной работы были получены навыки написания программы антиплагиата и облака тегов на языке C#. Был создан алгоритм Шинглов, найдено расстояние Левенштейна и выполнена кодировка в хэши.

Программы соответствуют заданиям и отвечают необходимым требованиям.