Министерство образования и науки РФ

ФГБОУ ВПО «Тамбовский государственный технический университет»

Кафедра « »

Лабораторная работа №4

по дисциплине «Теория информации»

«Алгоритм LZW»

Выполнил: студент гр. −31 Д. Ю.

Проверил: Н. В.

Тамбов,

***Цели и задачи.***

Разработать приложение, реализующее алгоритм LZW.

***Описание.***

Процесс сжатия выглядит следующим образом. Последовательно считываются символы входного потока и происходит проверка, существует ли в созданной таблице строк такая строка. Если такая строка существует, считывается следующий символ, а если строка не существует, в поток заносится код для предыдущей найденной строки, строка заносится в таблицу, а поиск начинается снова.

***Решение задачи****.*

Сжатие происходит исходя из результатов опроса по определенному алгоритму, примерно так: первый символ из входной строки есть в словаре? Если нет, то внести его в словарь и перейти к следующему символу. Иначе считать следующий символ и рассмотреть строку уже из двух символов как единое целое с целью выяснить, есть ли эта строка в словаре.

Восстановление строится на двух исходных – цепь сжатых данных и минимальный словарь, по которому происходит восстановление в дальнейшем всей кодовой строки. Работает принцип объединения, берем первую цифру входной последовательности. Она есть в словаре? Если нет, значит это ошибка словаря. Если есть, тогда считываем следующую цифру и объединяем значения в словаре (по ключу). Если такого значения нет, заносим в конец словаря, иначе считываем следующую цифру.

***Модули***

1

**public** clCompress**(**string str**)**

**{**

**if(**String**.**IsNullOrEmpty**(**str**))**

**return;**

**if(**StartDicton **!=** **null)**StartDicton**.**Clear**();**

var lst **=** **new** List**<**string**>();**

var index **=** 0**;**

\_list**.**Add**(new** KeyValuePair**<**string**,** int**>(**str**[**index**].**ToString**(),** index**));**

lst**.**Add**(**str**[**index**].**ToString**());**

//выпилить буквы без повтора, формируем начальный словарь;

**foreach** **(**char c **in** str**)**

**{**

**if** **(**lst**.**Contains**(**c**.**ToString**()))**

**continue;**

index**++;**

lst**.**Add**(**c**.**ToString**());**

\_list**.**Add**(new** KeyValuePair**<**string**,** int**>(**c**.**ToString**(),** index**));**

**}**

StartDicton **=** \_list**;**

Process**(**str**,** lst**);**

**}**

**internal** static List**<**KeyValuePair**<**string**,** int**>>** StartDicton **{** get**;** set**;** **}**

**private** void Process**(**string str**,** List**<**string **>** list**)**

**{**

Cod **=** **new** List**<**int**>();**

int index **=** 2**;**

var flag **=** **true;**

//взять первый символ сообщения и проверить его наличие в словаре

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

**{**

string tmp **=** str**[**i**].**ToString**();**

**if** **(**list**.**Contains**(**tmp**)** **&&** i **<** str**.**Length **-** 1**)**

flag **=** **true;**

**else** flag **=** **false;**

**if** **(**list**.**Contains**(**tmp**)** **&&** i **<** str**.**Length **-** 1**)**

**{**

//символ уже есть в словаре

//тогда берем tmp как два символа из str

tmp **=** str**.**Substring**(**i**,** index**);**

**while** **(**list**.**Contains**(**tmp**))**

**{**

index**++;**

**try**

**{**

flag **=** **true;**

tmp **=** str**.**Substring**(**i**,** index**);**

**}**

**catch** **(**Exception**)**

**{**

flag **=** **false;**

**break;**

**}**

**}**

**if** **(**flag**)**

**{**

list**.**Add**(**tmp**);**

**}**

Cod**.**Add**(**list**.**IndexOf**(**tmp**.**Substring**(**0**,** index **-** 1**)));**

**}**

**else**

**{**

//символа нет в словаре

**if** **(**flag**)**

**{**

list**.**Add**(**tmp**);**

**}**

Cod**.**Add**(**list**.**IndexOf**(**tmp**.**Substring**(**0**,** index **-** 1**)));**

**}**

//сместить положение курсора на позицию ++);

**if** **(**index **>** 2**)**

**{**

i **+=** **(**index **-** 2**);**

**}**

flag **=** **true;**

index **=** 2**;**

**}**

#region povtorы

//Проверить совпадения, если есть, то алгоритм ошибся

index **=** 0**;**

string**[]** Massiv **=** **new** string**[**list**.**Count **+** 1**];**

int ie **=** 0**,** j **=** 0**;**

**foreach** **(**string ts **in** list**)**

**{**

Massiv**[**j**]** **=** ts**;**

j**++;**

**for** **(**int e **=** 0**;** e **<** ie**;** e**++)**

**if** **(**ts**.**CompareTo**(**Massiv**[**e**])** **==** 0**)**

**{**//Если есть результ, то есть совпадение с текущим значением ts!

index**++;**//MessageBox.Show(ts);

Massiv**[**e**]** **=** **null;**//Занулим все совпадения

**}**

ie**++;**

**}**

#endregion

Repit **=** index**;**

Dictonary **=** list**;**

**}**

2

**public** clDecoder**(**string str**,** List**<**KeyValuePair**<**string**,** int**>>** dictonar**)**

**{**

**if(**dictonar**==null)** **return;**

var arr **=** str**.**Split**(**' '**);**

var flag **=** **false;**

//начальный словарь

var list **=** dictonar**.**Select**(**keyValuePair **=>** keyValuePair**.**Key**).**ToList**();**

**for** **(**int i **=** 0**;** i **<** arr**.**Length**-**1**;** i**++)**

**{**

var tmp **=** int**.**Parse**(**arr**[**i**]);**//0

**if** **(**SearchD**(**list**,** tmp**))**

**{**

var ses **=** list**[**tmp**];**//a

var indx **=** 1**;**

//проход по ячейкам

**while** **(**list**.**Contains**(**ses**))**

**{**

**if** **(**i **+** indx **>=** arr**.**Length**-**1**)**

**{**

flag **=** **true;**

**break;**

**}**

tmp **=** int**.**Parse**(**arr**[**i **+** indx**]);**//1

**if** **(**SearchD**(**list**,** tmp**))**

**{**

var tes **=** 0**;**

//проход по ЭЛЕМЕНТУ в ячейке

**while** **(**list**.**Contains**(**ses**))**

**{**

tes**++;**

**if** **(**list**[**tmp**].**Length **>=** tes**)**

ses **+=** list**[**tmp**].**Substring**(**0**,** tes**);**

**else** **break;**

**}**

**}**

indx**++;**

**}**

**if(**flag**)** **break;**

list**.**Add**(**ses**);**

**}**

**else**

**{**

//list.Add(tmp);

**}**

**}**

Decod **=** list**;**

**}**

**private** bool SearchD**(**List**<**string **>** list**,** int i**)**

**{**

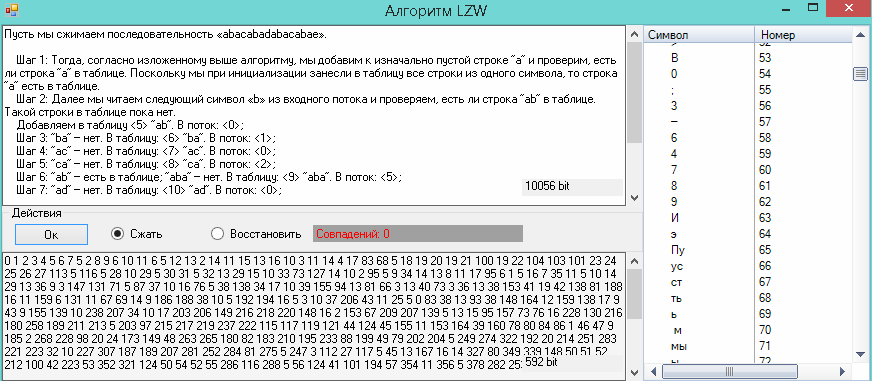
**return** i **<** list**.**Count**;**

**}**

***Выбор языка программирования***

Язык C#, самый легкий и больше всего изученный мною.

***Тестирование программы***

******