Учреждение образования

«Белорусский государственный технологический университет»

**Защита информации и надежность информационных систем**

**Лабораторная работа №10**

СЖАТИЕ/РАСПАКОВКА ДАННЫХ МЕТОДОМ ЛЕМПЕЛЯ − ЗИВА Выполнил:

Студент 3 курса 2 группы ФИТ

Максимова Вера Владимировна

Проверила:

Ржеутская Надежда Викентьевна

**2022 г.**

Цель: приобретение практических навыков использования метод Лемпеля − Зива (Lempel-Ziv) для сжатия/распаковки данных.

Задачи:

1. Закрепить теоретические знания по алгебраическому описанию и использованию методов сжатия/распаковки (архивации/ разархивации) данных на основе метода Лемпеля − Зива.

2. Разработать приложение для реализации метода Лемпеля − Зива.

3. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LAB\_9

{

public class LZMessageList

{

private const string Bin = "01";

private const string Quad = "0123";

private const string Oct = "01234567";

private const string Hex = "0123456789ABCDEFG";

const string Step = "Step ";

const string Separator = "----------------------------------------------------";

private char[] Dictionary;

private char[] Buffer;

public int MessageLength;

public Mode MessageMode { get; private set; }

private List<LZMessage> LZMessages;

public int Index { get; private set; }

public int Count

{

get

{

return LZMessages.Count();

}

}

private string FilePath;

public LZMessageList(Mode mode, int dictionaryLength, int bufferLength, string path)

{

MessageMode = mode;

Index = 0;

Dictionary = new char[dictionaryLength];

Buffer = new char[bufferLength];

LZMessages = new List<LZMessage>();

FilePath = path;

int logn1 = (int)Math.Ceiling(Math.Log(dictionaryLength, (double)mode));

int logn2 = (int)Math.Ceiling(Math.Log(bufferLength, (double)mode));

MessageLength = logn1 + logn2 + 1;

}

public void AddText(string text)

{

var sf = File.Create(FilePath);

sf.Close();

LZMessages.Clear();

int textIndex = 0;

Queue<char> queue = new Queue<char>(text);

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

Dictionary[i] = '0';

for (int i = 0; i < Buffer.Length; i++, textIndex++)

{

if (queue.Count != 0)

Buffer[i] = queue.Dequeue();

else

Buffer[i] = '-';

}

Console.WriteLine("----------------------------------------------------");

Console.WriteLine("Encryption");

Console.WriteLine("------------------------------------------------------------");

Console.WriteLine(new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, "Encryption");

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

int numlen = (MessageLength - 1) / 2;

int windowlen = Dictionary.Length + Buffer.Length;

int index = 0;

while (Buffer[0] != '-')

{

string window = new string(Dictionary) + new string(Buffer);

int bufstart = Dictionary.Length;

int len = 0;

int place = 0;

for (int i = 1; i < Buffer.Length; i++)

{

string symbols = window.Substring(bufstart, i);

if (symbols.Last() == '-') break;

int find = window.IndexOf(symbols);

if (find != -1 && find < bufstart)

{

place = find;

len = i;

}

else

break;

}

char s = window[bufstart + len];

if (s == '-')

{

len = 0;

place = 0;

s = window[bufstart + len];

}

string p = Convert(place);

string q = Convert(len);

while (p.Length < numlen)

p = p.Insert(0, "0");

while (q.Length < numlen)

q = q.Insert(0, "0");

var message = new LZMessage(MessageMode, MessageLength, p + q + s);

Add(message);

Console.WriteLine("------------------------------------------------------------");

Console.WriteLine(new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

Console.WriteLine(LZMessages.Last().ToString());

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, Step + (index + 1).ToString());

Additions.EnterLineToFile(FilePath, new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

Additions.EnterLineToFile(FilePath, LZMessages.Last().ToString());

len++;

index++;

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

{

if (i + len < Dictionary.Length)

Dictionary[i] = Dictionary[i + len];

else

Dictionary[i] = Buffer[i + len - bufstart];

}

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

{

if (i + len < Buffer.Length)

Buffer[i] = Buffer[i + len];

else

{

if (queue.Count != 0)

Buffer[i] = queue.Dequeue();

else

Buffer[i] = '-';

}

}

}

Console.WriteLine("------------------------------------------------------------");

Console.WriteLine(new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

Console.WriteLine("------------------------------------------------------------");

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, new string(Dictionary) + " " + new string(Buffer) + " " + new string(queue.ToArray()));

Additions.EnterLineToFile(FilePath, Separator);

}

public string GetText()

{

string result = "";

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

Dictionary[i] = '0';

Console.WriteLine("----------------------------------------------------");

Console.WriteLine("Decryption");

Console.WriteLine("----------------------------------------------------");

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, "Decryption");

Additions.EnterLineToFile(FilePath, Separator);

Console.WriteLine(new string(Dictionary) + " " + GetMessagesString(0, true));

int numlen = (MessageLength - 1) / 2;

for (int i = 0; i < LZMessages.Count; i++)

{

Console.WriteLine("----------------------------------------------------");

Console.WriteLine(new string(Dictionary) + " " + GetMessagesString(i, true));

Console.WriteLine(result.GetSplitedString((int)Math.Log(256, (int)MessageMode)));

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, Step + (i + 1).ToString());

Additions.EnterLineToFile(FilePath, new string(Dictionary) + " " + GetMessagesString(i, true));

Additions.EnterLineToFile(FilePath, result.GetSplitedString((int)Math.Log(256, (int)MessageMode)));

var mes = LZMessages[i];

string dict = new string(Dictionary);

string buf = mes.Message;

int p = buf.Substring(0, numlen).ArbitraryToDecimalSystem((int)MessageMode);

int q = buf.Substring(numlen, numlen).ArbitraryToDecimalSystem((int)MessageMode);

char s = buf.Last();

string part = "";

if (p + q > dict.Length)

{

int razn = Math.Abs(dict.Length - p - q);

int dictcount = q - razn;

string dictbuf;

dictbuf = dict.Substring(p, dictcount);

part += dictbuf;

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

{

part += dictbuf[j % dictbuf.Length];

}

part += s;

result += part;

}

else

{

part = dict.Substring(p, q) + s;

result += part;

}

int offset = part.Length;

int partIndex = 0;

for (int j = 0; j < Dictionary.Length; j++)

{

if (j + offset < Dictionary.Length)

{

Dictionary[j] = Dictionary[j + offset];

}

else

{

Dictionary[j] = part[partIndex];

partIndex++;

}

}

}

Console.WriteLine("----------------------------------------------------");

Console.WriteLine(result.GetSplitedString((int)Math.Log(256, (int)MessageMode)));

Additions.EnterLineToFile(FilePath, Separator);

Additions.EnterLineToFile(FilePath, Step + "Last");

Additions.EnterLineToFile(FilePath, result.GetSplitedString((int)Math.Log(256, (int)MessageMode)));

return result;

}

private void Add(LZMessage message)

{

if (message.MessageMode == MessageMode)

LZMessages.Add(message);

else

throw new Exception("Modes of messages and list don't intercept");

}

private string Convert(int num)

{

string result = "";

result = num.DecimalToArbitrarySystem((int)MessageMode);

return result;

}

private LZMessage GetNextMessage()

{

if (Index < Count)

{

Index++;

return LZMessages[Index];

}

else

throw new ArgumentOutOfRangeException();

}

public string GetMessagesString(int from, bool spaces)

{

string result = "";

for (int i = from; i < LZMessages.Count; i++)

{

if (spaces)

result += LZMessages[i].ToString();

else

result += LZMessages[i].Message;

}

return result;

}

public override string ToString()

{

string result = "";

foreach (var mes in LZMessages)

result += mes.ToString();

return result;

}

}

}