Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ивановский государственный энергетический университет имени В.И. Ленина»

Кафедра программного обеспечения компьютерных систем

Отчет по лабораторной работе №1

Шифрование простой подстановкой и перестановкой.

Генераторы псевдослучайных чисел, гаммирование.

Выполнила студентка гр. 4-41 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Митюшина А.А.

Проверил

к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Игнатьев Е.Б.

Иваново, 2019

**Задание**

Напишите программы шифрования и расшифрования с использованием файла со стихотворением. Каждую букву открытого текста шифруют парой чисел – номером строки, где встречается эта буква, и номером буквы в ней. Поиск следующего символа начнется от только что найденного. Открытый, ключевой и зашифрованный тексты должны находиться в текстовых файлах.

**Листинг**

class EncoderCoder

{

int line = 1;

int position = 1;

string plainTextPath = @".\text\plaintext.txt";

string cipherTextPath = @".\text\ciphertext.txt";

string keyPath = @".\text\key.txt";

string plain2TextPath = @".\text\plaintext2.txt";

public int readSymbol (StreamReader streamReaderPlanText)

{

if (streamReaderPlanText.Peek() != -1)

{

return streamReaderPlanText.Read();

}

else

{

return -1;

}

}

public string getCode(int? currentSymbol, StreamReader streamReaderKey)

{

int peekSymbol = streamReaderKey.Peek();

while (streamReaderKey.Peek() != currentSymbol)

{

switch (peekSymbol)

{

case -1:

//streamReaderKey = new StreamReader(keyPath, Encoding.Default);

streamReaderKey.BaseStream.Position = 0;

line = 1;

position = 1;

peekSymbol = streamReaderKey.Read();

break;

case 10:

line++;

position = 1;

streamReaderKey.Read();

peekSymbol = streamReaderKey.Peek();

break;

default:

streamReaderKey.Read();

position++;

peekSymbol = streamReaderKey.Peek();

break;

}

}

string result = line.ToString() + ' ' + position.ToString() + ' ';

streamReaderKey.Read();

position++;

return result;

}

public void encrypt ()

{

StreamReader streamReaderPlanText = new StreamReader(plainTextPath, Encoding.Default);

StreamWriter streamWriterCipherTextPath = new StreamWriter(cipherTextPath);

StreamReader streamReaderKey = new StreamReader(keyPath, Encoding.Default);

int currentSymbol = readSymbol(streamReaderPlanText);

while (currentSymbol !=-1)

{

streamWriterCipherTextPath.Write(getCode(currentSymbol, streamReaderKey));

currentSymbol = readSymbol(streamReaderPlanText);

}

streamReaderPlanText.Close();

streamWriterCipherTextPath.Close();

streamReaderKey.Close();

}

public void toDecipher ()

{

StreamReader streamReaderCipherText = new StreamReader(cipherTextPath, Encoding.Default);

StreamReader streamReaderKey = new StreamReader(keyPath, Encoding.Default);

StreamWriter streamWriterplain2TextPath = new StreamWriter(plain2TextPath);

int currentSymbol = readSymbol(streamReaderCipherText);

int[] symbol = new int[2];

string line = "";

string pos = "";

int countObj = 0;

while (currentSymbol != -1)

{

if (currentSymbol == 32)

{

switch (countObj)

{

case 0:

countObj++;

break;

case 1:

streamWriterplain2TextPath.Write(getSymbol(line, pos, streamReaderKey));

line = "";

pos = "";

countObj = 0;

break;

}

}

else

{

switch (countObj)

{

case 0:

int sm = currentSymbol - 48;

line = line + sm.ToString();

break;

case 1:

int p = currentSymbol - 48;

pos = pos + p.ToString();

break;

}

}

currentSymbol = readSymbol(streamReaderCipherText);

}

streamReaderCipherText.Close();

streamReaderKey.Close();

streamWriterplain2TextPath.Close();

}

private char getSymbol(string lineString, string posString, StreamReader streamReaderKey)

{

int line = Convert.ToInt32(lineString);

int pos = Convert.ToInt32(posString);

int currentLine=1, currentPos=1;

int currentSymbol=-1;

streamReaderKey = new StreamReader(keyPath, Encoding.Default);

//streamReaderKey.BaseStream.Position = 0;

while (currentLine<line)

{

if (streamReaderKey.Peek() == 10)

{

currentLine++;

}

streamReaderKey.Read();

}

while (currentPos <pos)

{

currentPos++;

streamReaderKey.Read();

}

currentSymbol = streamReaderKey.Read();

return Convert.ToChar(currentSymbol) ;

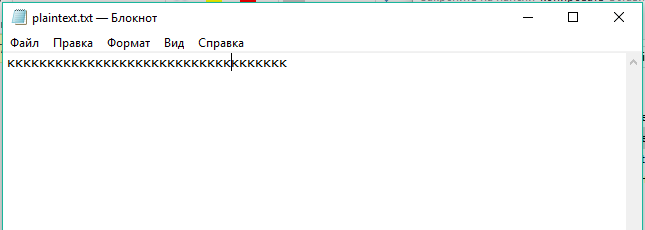
}

}

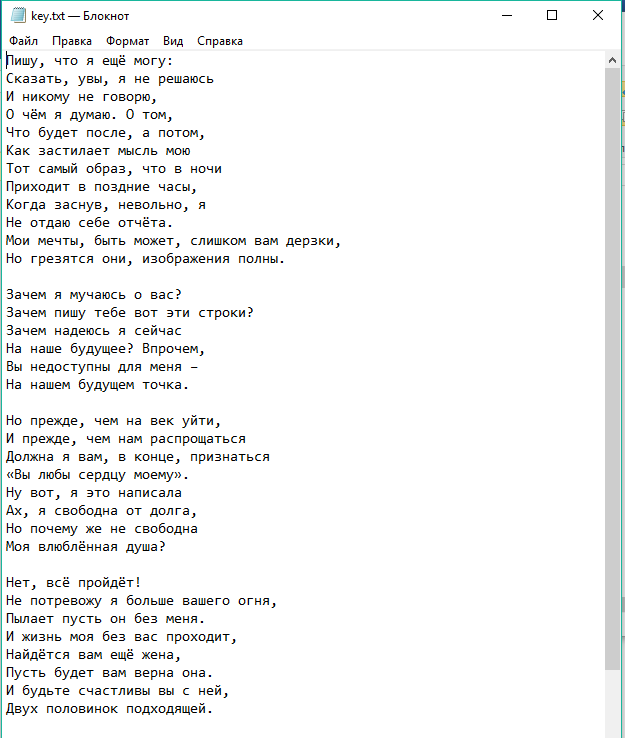
**Тестирование**

***Пример 1.***

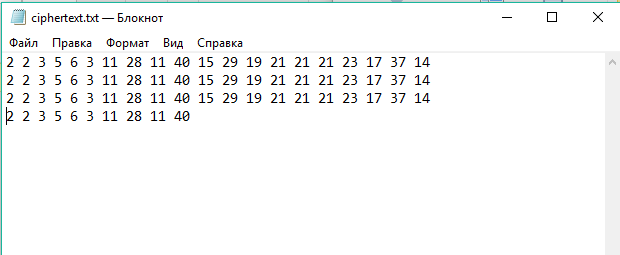
Открытый текст.



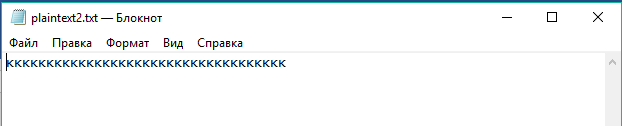
Ключ.



Зашифрованный текст (перенос символов только для наглядности результата).



Раскодированный текст.



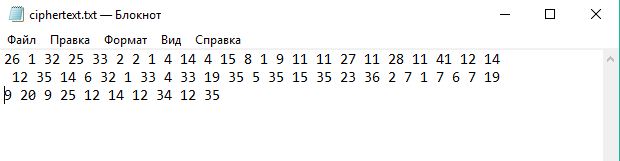
***Пример 2.***

Открытый текст.



Ключ (смотри пример 1).

Зашифрованный текст



Раскодированный текст

