#### Министерство науки и высшего образования Российской Федерации

#### Федеральное государственное бюджетное образовательное учреждение

#### высшего образования

#### «Владимирский государственный университет

#### имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

Лабораторная работа №6

по дисциплине

"Администрирование и безопасность программных информационных систем"

Выполнил:

ст. гр. ПРИ-117

Куппе Р.О

Принял:

Спирин И.В

Владимир, 2020 г.

## **Цель работы**

Освоить на практике применение алгоритма шифрования с помощью аналитических преобразований.

**Выполнение работы**

Репозиторий на github: <https://github.com/Kuppuch/Matrix-cipher.git>

Суть алгоритма шифрования с помощью аналитических преобразований в том, чтобы матрицу (которая является ключом) умножить на вектор, который скалывается из индексов букв текста.

Дешиврование отличается от шифрования только матрицей. В качестве ключа дешифровки следует взять обратную матрицу.

Обратная матрица была высчитана сторонними ресурсами, так как считать определитель для матрицы 4X4 программно крайне затратнои намного тяжелее чем для матрицы 3X3.

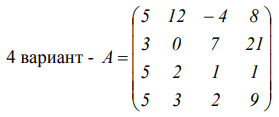


Рисунок 1 – Матрица-ключ



Рисунок 2 – Обратная матрица

Код программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Numerics;

using System.Text;

using System.Threading.Tasks;

namespace MatrixCode2.\_0 {

class Program {

static char[] alpha = new char[] {'а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы',

'ь', 'э', 'ю', 'я' }; // 31 буква

static string text = "Помехоустойчивое кодирование это кодирование с возможностью восстановления потерянных или ошибочно принятых данных";

static void Main(string[] args) {

double[,] matrix = new double[,] { { 5, 12, -4, 8 }, { 3, 0, 7, 21 }, { 5, 2, 1, 1 }, { 5, 3, 2, 9 } };

double[,] umatrix = new double[,] { { -0.16, -0.29, -0.321, 0.855 }, { 0.294, 0.448, 0.838, -1.401 }, { 0.286, 0.697, 1.323, -2.027 }, { -0.0725, -0.143, -0.395, 0.553 } };

text = text.Replace(" ", "").ToLower();

int[] vector = new int[4];

string code = "";

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

if (i % vector.Length == 0 && i > 0)

code += Multiply(matrix, vector);

if (i == 0) {

vector[i] = Array.IndexOf(alpha, text[i]);

continue;

}

vector[i % vector.Length] = Array.IndexOf(alpha, text[i]);

}

code = code.Trim();

string uncode = "";

Console.WriteLine("Зашифрованный текст:");

Console.WriteLine(code +"\n");

int[] dparse = DeCrypt(code);

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

if (i % vector.Length == 0 && i > 0)

uncode += Multiply(umatrix, vector);

vector[i % vector.Length] = dparse[i];

}

dparse = DeCrypt(uncode.Trim());

uncode = "";

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

uncode += alpha[dparse[i]];

}

Console.WriteLine("Расшифрованный текст:");

Console.WriteLine(uncode);

Console.ReadKey();

}

public static string Multiply(double[,] matrix, int[] vector) {

double result;

string answ = "";

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

result = 0;

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

result += matrix[i, j] \* vector[j];

}

answ += Math.Round(result) + " ";

}

// Для просмотра векторов расскоментировать следующую строку

// Console.WriteLine("Тут д.б. шифрованная строка " + answ);

return answ;

}

public static int[] DeCrypt(string code) {

string[] parse = code.Split(' ');

int[] dparse = new int[parse.Length];

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

dparse[i] = Convert.ToInt32(parse[i]);

}

return dparse;

}

}

}

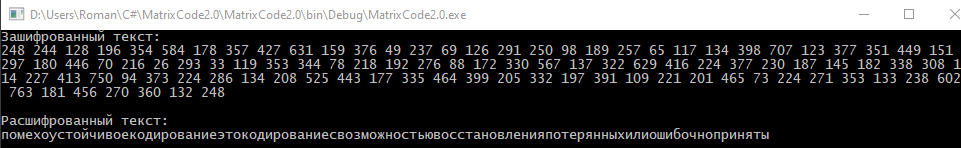


Рисунок 3 – Результат работы программы

## **Вывод**

В ходе выполнения лабораторной работы было освоено на практике применение алгоритма шифрования с помощью аналитических преобразований.