

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационной безопасности

**ОТЧЕТ**

по лабораторной работе №1

на тему: «**Простейшие криптографические системы**»

по дисциплине «Информационная безопасность»

Выполнили: Кожухова О.А.

Шифр: 170582

Шорин В.Д.

Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверил: Еременко В.Т.

Отметка о зачете: \_\_\_\_\_

Дата «\_\_\_\_» \_\_\_\_\_ 2021г.

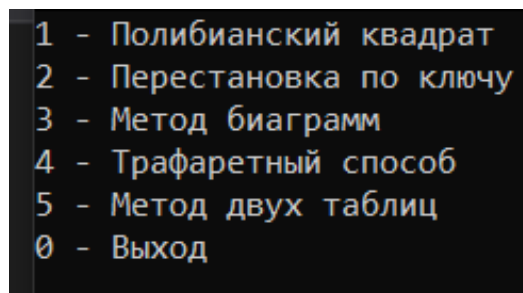
Орел, 2021 г.

## Задание

Напишите программу, реализующую метод шифровки сообщения:

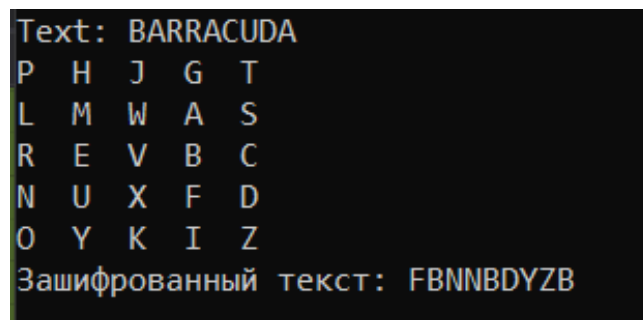
1. – основанный на полибианском квадрате;
2. – путем перестановки по ключу;
3. – методом биграмм;
4. – трафаретным способом;
5. – методом двух таблиц.

## Ход работы



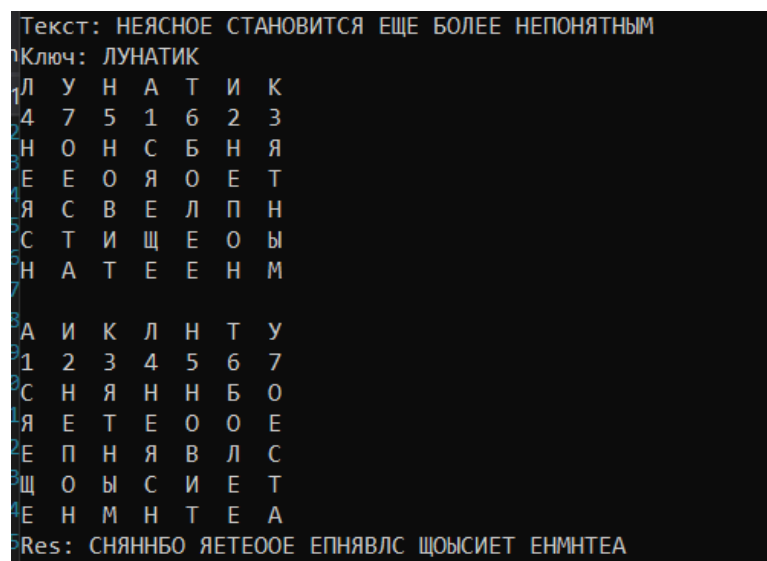
```
1 - Полибианский квадрат
2 - Перестановка по ключу
3 - Метод биграмм
4 - Трафаретный способ
5 - Метод двух таблиц
0 - Выход
```

Рисунок 1 – Меню программы



```
Text: BARRACUDA
P H J G T
L M W A S
R E V B C
N U X F D
O Y K I Z
Зашифрованный текст: FBNNBDYZB
```

Рисунок 2 – Работа программы «Полибианский квадрат»



```
Текст: НЕЯСНОЕ СТАНОВИТСЯ ЕЩЕ БОЛЕЕ НЕПОНЯТНЫМ
Ключ: ЛУНАТИК
1 Л У Н А Т И К
2 4 7 5 1 6 2 3
3 Н О Н С Б Н Я
4 Е Е О Я О Е Т
5 Я С В Е Л П Н
6 С Т И Щ Е О Ы
7 Н А Т Е Е Н М
8 А И К Л Н Т У
9 1 2 3 4 5 6 7
0 С Н Я Н Н Б О
1 Я Е Т Е О О Е
2 Е П Н Я В Л С
3 Щ О Ы С И Е Т
4 Е Н М Н Т Е А
5 Res: СНЯННБО ЯЕТЕООЕ ЕПНЯВЛС ЩОЫСИЕТ ЕНМНТЕА
```

Рисунок 3 – Работа программы «Перестановка по ключу»

```

Текст: ПУСТЬ КОНСУЛЫ БУДУТ БДИТЕЛЬНЫ
Ключ: РЕСПУБЛИКА
пустьконсулыбудутбдительны
р е с п у б
л и к а в г
д ж з м н о
т ф х ц ч ш
щ ь ы э ю я
Res: уб рх ыи до пб кщ рб нр шр жл фр ищ зю

```

Рисунок 4 – Работа программы «Метод биграмм»

```

Text: ПРИЕЗЖАЮ ШЕСТОГО
з т п
о ж ш р
е и г а
е с ю о
Res: зтп ожшреигаесю

```

Рисунок 5 – Работа программы «Трафаретный способ»

```

Текст: ПУСТЬ КОНСУЛЫ БУДУТ БДИТЕЛЬНЫ
р ж л ё п в ы й ж ч
ч с к э у ю р щ з
ш з в д к ё д г ф
е ы ь я т е ь о ь м
! ф ь и м я п а ш б
н й а о г н т и х с
х щ ц ю б л э ц у !
Res: ё!сть когнч!рю!в!ьзцффхелеиьр

```

Рисунок 6 – Работа программы «Метод двух таблиц»

## Код

### «Program.cs»

```

using System;

namespace IS_L_1
{
    class Program
    {
        static void Main(string[] args)
        {
            while(true)
            {
                Console.Clear();
                Console.WriteLine("1 - Полибианский квадрат");
                Console.WriteLine("2 - Перестановка по ключу");
                Console.WriteLine("3 - Метод биаграмм");
            }
        }
    }
}

```

```

Console.WriteLine("4 - Трафаретный способ");
Console.WriteLine("5 - Метод двух таблиц");
Console.WriteLine("0 - Выход");

int res = Convert.ToInt32(Console.ReadLine());

switch (res)
{
    case 1:
    {
        Console.Clear();

        //Console.Write("Текст: ");
        //string str = Console.ReadLine().ToUpper();
        string text = "BARRACUDA";
        Console.WriteLine($"Текст: {text}");
        task1 t1 = new task1();

        string result = t1.EncryptionByPolibiusSquare(text);
        Console.WriteLine($"Зашифрованный текст: {result}");
        Console.ReadLine();
        break;
    }
    case 2:
    {
        Console.Clear();
        task2 t2 = new task2();

        //Console.Write("Текст: ");
        //string text = Console.ReadLine().ToLower();
        //Console.Write("Ключ: ");
        //string key = Console.ReadLine().ToLower();

        string text = "НЕЯСНОЕ СТАНОВИТСЯ ЕЩЕ БОЛЕЕ НЕПОНЯТНЫМ";
        string key = "ЛУНАТИК";
        Console.WriteLine($"Текст: {text} \nКлюч: {key}");
        string result = t2.PermutationByKey(text, key);
        Console.WriteLine($"Res: {result}");
        Console.ReadLine();
        break;
    }
    case 3:
    {
        Console.Clear();
        task3 t3 = new task3();

        //Console.Write("Текст: ");
        //string text = Console.ReadLine();
        //Console.Write("Ключ: ");
        //string key = Console.ReadLine();

        string text = "ПУСТЬ КОНСУЛЫ БУДУТ БДИТЕЛЬНЫ ";
        string key = "РЕСПУБЛИКА";
        Console.WriteLine($"Текст: {text} \nКлюч: {key}");

        if (text.Length % 2 != 0)
        {
            Console.WriteLine("В тексте должно быть четное число
символов");

            break;
        }
        string result = t3.Bigram(text, key);
        Console.WriteLine($"Res: {result}");
        Console.ReadLine();
        break;
    }
}

```

```

    }
    case 4:
    {
        Console.Clear();

        task4 t4 = new task4();
        string text = "ПРИЕЗЖАЮ ШЕСТОГО";
        Console.WriteLine($"Text: {text}");

        string result = t4.CardanoGrid(text);
        Console.WriteLine($"Res: {result}");
        Console.ReadLine();
        break;
    }
    case 5:
    {
        Console.Clear();
        task5 t5 = new task5();
        string text = "ПУСТЬ КОНСУЛЫ БУДУТ БДИТЕЛЬНЫ ";
        Console.WriteLine($"Текст: {text}");

        string result = t5.TwoTables("ПУСТЬ КОНСУЛЫ БУДУТ БДИТЕЛЬНЫ ");
        Console.WriteLine($"Res: {result}");
        Console.ReadLine();
        break;
    }
    case 0:
        return;
    default:
        Console.WriteLine("Нет такой команды");
        break;
    }
}

public static void PrintTable(string[,] table)
{
    int rows = table.GetUpperBound(0) + 1;
    int cols = table.GetUpperBound(1) + 1;

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            Console.Write($"{table[i, j]} ");
        }
        Console.WriteLine();
    }
}

public static void PrintTable(char[,] table)
{
    int rows = table.GetUpperBound(0) + 1;
    int cols = table.GetUpperBound(1) + 1;

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            Console.Write($"{table[i, j]} ");
        }
        Console.WriteLine();
    }
}
}
}
}

```

## «task1.cs»

```
using System;
using System.Collections.Generic;
using System.Text;

namespace IS_L_1
{
    class task1
    {
        private string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        private static int side = 5;
        private char[,] square;
        public string EncryptionByPolibiusSquare(string text)
        {
            string outputText = "";
            square = GetSquare();
            Program.PrintTable(square);

            string newText = text.Replace('Q', 'O');

            for (int i = 0; i < newText.Length; i++)
            {
                if (FindSymbol(square, newText[i], out int columnIndex, out int
rowIndex))
                {
                    int newRowIndex = rowIndex == square.GetUpperBound(1) ? 0 : rowIndex
+ 1;

                    outputText += square[newRowIndex, columnIndex].ToString();
                }
            }

            return outputText;
        }

        private bool FindSymbol(char[,] table, char symbol, out int column, out int row)
        {
            int l = table.GetUpperBound(0) + 1;
            for (int i = 0; i < l; i++)
            {
                for (int j = 0; j < l; j++)
                {
                    if (table[i, j] == symbol)
                    {
                        row = i;
                        column = j;
                        return true;
                    }
                }
            }

            row = -1;
            column = -1;
            return false;
        }

        private char[,] GetSquare()
        {
            string newAlphabet = alphabet;
            char[,] square = new char[side, side];
            Random random = new Random();

            for (int i = 0; i < side; i++)
```

```

        {
            for (int j = 0; j < side; j++)
            {
                int value = random.Next(0, newAlphabet.Length - 1);
                square[i, j] = newAlphabet[value];
                newAlphabet = newAlphabet.Remove(value, 1);
            }
        }

        return square;
    }

    public string DecryptionByPolibiusSquare(string text)
    {
        string outputText = "";
        int m = text.Length;

        for (int i = 0; i < m; i++)
        {
            if (FindSymbol(square, text[i], out int columnIndex, out int rowIndex))
            {
                int newRowIndex = rowIndex == 0 ? square.GetUpperBound(1) : rowIndex - 1;
                outputText += square[newRowIndex, columnIndex].ToString();
            }
        }

        return outputText;
    }
}

```

## «task2.cs»

```

using System;
using System.Collections.Generic;
using System.Text;

namespace IS_L_1
{
    class task2
    {
        public string PermutationByKey(string text, string key)
        {
            int keyLength = key.Length;
            string[,] table = new string[keyLength, keyLength];
            string[,] tableResult = new string[keyLength, keyLength];
            Dictionary<string, int> symbols = new Dictionary<string, int>();

            char[] charArray = key.ToCharArray();
            Array.Sort(charArray);
            string keySorted = new string(charArray);

            for (int i = 0; i < keySorted.Length; i++)
            {
                symbols.Add(keySorted[i].ToString(), i + 1);
            }

            for (int i = 0; i < keyLength; i++)
            {
                table[0, i] = key[i].ToString();
                tableResult[0, i] = keySorted[i].ToString();
            }
        }
    }
}

```

```

        table[1, i] = symbols[key[i].ToString()].ToString();
        tableResult[1, i] = (i + 1).ToString();
    }

    int k = 0;
    for (int i = 0; i < keyLength; i++)
    {
        for (int j = 2; j < keyLength; j++)
        {
            if (k > text.Length) break;
            while (text[k].ToString() == " ") k++;
            table[j, i] = text[k].ToString();
            k++;
        }
    }

    for (int i = 0; i < keyLength; i++)
    {
        k = Convert.ToInt32(table[1, i]) - 1;
        for (int j = 2; j < keyLength; j++)
        {
            tableResult[j, k] = table[j, i];
        }
    }

    string result = "";
    for (int i = 2; i < keyLength; i++)
    {
        for (int j = 0; j < keyLength; j++)
        {
            result += tableResult[i, j];
        }
        result += " ";
    }

    Program.PrintTable(table);
    Console.WriteLine();
    Program.PrintTable(tableResult);

    return result;
}
}
}

```

### « task3.cs »

```

using System;
using System.Collections.Generic;
using System.Text;

namespace IS_L_1
{
    class task3
    {
        private string alphabet = "абвгдежзиклмнопрстуфхцщщыэюя";
        private int rows = 5;
        private int cols = 6;
        public string Bigram(string text, string key)
        {
            string[,] table = new string[rows, cols];
            string newText = text.Replace(" ", "");

            key = Uniq(key);

```



```

newText = newText.ToLower();
key = key.ToLower();
alphabet = alphabet.ToLower();

for (int i = 0; i < newText.Length; i++)
{
    if (newText[i] == 'ë')
    {
        newText.Replace(newText[i], 'e');
    }
    else if (newText[i] == 'ё')
    {
        newText.Replace(newText[i], 'e');
    }
    else if (newText[i] == 'ь')
    {
        newText.Replace(newText[i], 'b');
    }
}

string tableAlphabet = key;
foreach (var symbol in alphabet)
{
    if (!key.Contains(symbol))
    {
        tableAlphabet += symbol;
    }
}

int index = 0;
for (int i = 0; i < rows; i++)
{
    for (int j = 0; j < cols; j++)
    {
        table[i, j] = tableAlphabet[index].ToString();
        index++;
    }
}

Console.WriteLine(newText);
string encryptedString = "";
for (int i = 0; i < newText.Length; i += 2)
{
    int rowFirst = RowNumber(table, newText[i]);
    int rowSecond = RowNumber(table, newText[i + 1]);
    int colFirst = ColumnNumber(table, newText[i]);
    int colSecond = ColumnNumber(table, newText[i + 1]);

    if (rowFirst == rowSecond)
    {
        string s1 = (colFirst == cols - 1) ? table[rowFirst, 0] :
table[rowFirst, colFirst + 1];
        string s2 = (colSecond == cols - 1) ? table[rowSecond, 0] :
table[rowSecond, colSecond + 1];
        encryptedString += s1 + s2;
    }
    else if (colFirst == colSecond)
    {
        Console.WriteLine($"{newText[i]}: {rowFirst},{colFirst}\n{newText[i +
1]}: {rowSecond},{colSecond}");
        string s1 = (rowFirst == rows - 1) ? table[0, colFirst] :
table[rowFirst + 1, colFirst];
        string s2 = (rowSecond == rows - 1) ? table[0, colSecond] :
table[rowSecond + 1, colSecond];
        encryptedString += s1 + s2;
    }
}

```

```

        }
        else
        {
            encryptedString += table[rowFirst, colSecond];
            encryptedString += table[rowSecond, colFirst];
        }

        encryptedString += ' ';
    }

    Program.PrintTable(table);

    return encryptedString;
}

private string Uniq(string s)
{
    List<char> used = new List<char>();
    StringBuilder uniq = new StringBuilder();
    foreach (char i in s)
    {
        if (used.IndexOf(i) == -1)
        {
            used.Add(i);
            uniq.Append(i);
        }
    }
    return uniq.ToString();
}

private int RowNumber(string[,] table, char s)
{
    int count = 0;
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            if (table[i, j] == s.ToString())
            {
                return i;
            }
        }
    }
    return 0;
}

private int ColumnNumber(string[,] table, char s)
{
    int count = 0;
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            if (table[i, j] == s.ToString())
            {
                return j;
            }
        }
    }
    return 0;
}
}
}

```

## « task4.cs »

```
using System;
using System.Collections.Generic;
using System.Text;

namespace IS_L_1
{
    class task4
    {
        public string CardanoGrid(string text)
        {
            const int SIZE = 4;
            string[,] grid = new string[SIZE, SIZE] {{ "#", "#", " ", "#" },
                                                    { "#", "#", "#", " " },
                                                    { "#", " ", "#", "#" },
                                                    { " ", "#", "#", "#" } };

            string[,] table = new string[SIZE, SIZE];

            string newText = text.ToLower();
            while (newText.Length < 16) newText += ' ';

            int index = 0;
            // Прямой обход
            for (int i = 0; i < SIZE; i++)
            {
                for (int j = 0; j < SIZE; j++)
                {
                    if (grid[i, j] == " ")
                    {
                        table[i, j] = newText[index].ToString();
                        index++;
                    }
                }
            }

            //90 градусов
            for (int i = 0; i < SIZE; i++)
            {
                for (int j = 0; j < SIZE; j++)
                {
                    if (grid[SIZE - j - 1, i] == " ")
                    {
                        table[i, j] = newText[index].ToString();
                        index++;
                    }
                }
            }

            //180 градусов
            for (int i = 0; i < SIZE; i++)
            {
                for (int j = 0; j < SIZE; j++)
                {
                    if (grid[SIZE - i - 1, SIZE - j - 1] == " ")
                    {
                        table[i, j] = newText[index].ToString();
                        index++;
                    }
                }
            }

            //270 градусов
            for (int i = 0; i < SIZE; i++)
```

```

        {
            for (int j = 0; j < SIZE; j++)
            {
                if (grid[j, SIZE - i - 1] == " ")
                {
                    table[i, j] = newText[index].ToString();
                    index++;
                }
            }
        }

        string res = "";
        for (int i = 0; i < SIZE; i++)
        {
            for (int j = 0; j < SIZE; j++)
            {
                res += table[i, j];
            }
        }

        Program.PrintTable(table);

        return res;
    }
}

```

### « task5.cs »

```

using System;
using System.Collections.Generic;
using System.Text;

namespace IS_L_1
{
    class task5
    {
        private string alphabet = "абвгдеёжзийклмнопрстуфхцчщъыэюя !1";
        private static int rows = 7;
        private static int cols = 5;
        private char[,] table1 = new char[rows, cols];
        private char[,] table2 = new char[rows, cols];

        public string TwoTables(string text)
        {
            string resultText = "";
            if (text.Length % 2 != 0)
            {
                text += ' ';
            }
            FillTables();

            text = text.ToLower();
            int length = text.Length / 2;
            int k = 0;
            char[,] bigrams = new char[length, 2];
            char[,] kriptobigrams = new char[length, 2];

            for (int i = 0; i < length; i++)
            {
                for (int j = 0; j < 2; j++)
                {
                    bigrams[i, j] = text[k];

```

```

        k++;
    }
}

int step = 0;
while (step < length)
{
    Indexes indexes1 = FindIndexes(bigrams[step, 0], table1);
    Indexes indexes2 = FindIndexes(bigrams[step, 1], table2);

    kryptoBigrams[step, 0] = table1[indexes1.I, indexes2.J];
    kryptoBigrams[step, 1] = table2[indexes2.I, indexes1.J];

    step++;
}

for (int i = 0; i < length; i++)
{
    for (int j = 0; j < 2; j++)
    {
        resultText += kryptoBigrams[i, j].ToString();
    }
}

PrintTables(table1, table2);

return resultText;
}

private void FillTables()
{
    string newAlphabet = alphabet;
    Random random = new Random();
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            int value = random.Next(0, newAlphabet.Length - 1);
            table1[i, j] = newAlphabet[value];
            newAlphabet = newAlphabet.Remove(value, 1);
        }
    }

    newAlphabet = alphabet;
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            int value = random.Next(0, newAlphabet.Length - 1);
            table2[i, j] = newAlphabet[value];
            newAlphabet = newAlphabet.Remove(value, 1);
        }
    }
}

private static void PrintTables(char[,] table1, char[,] table2)
{
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            Console.Write($"{table1[i, j]} ");
        }
        Console.WriteLine("");
    }
}

```

```

        for (int j = 0; j < cols; j++)
        {
            Console.Write($"{table2[i, j]} ");
        }
        Console.WriteLine();
    }
}

private Indexes FindIndexes(char symbol, char[,] matrix)
{
    Indexes cortege = new Indexes();

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            if (symbol == matrix[i, j])
            {
                cortege.I = i;
                cortege.J = j;
                return cortege;
            }
        }
    }

    return null;
}

private class Indexes
{
    public int I { get; set; }
    public int J { get; set; }
    public Indexes() { }
}
}

```