УО «Белорусский государственный университет информатики и радиоэлектроники»
Кафедра ПОИТ

Отчет по лабораторной работе №2
по предмету
Теория Информации

Вариант 14

Выполнил:
Бетеня К.С.

Проверил:
Болтак С.В.

Группа 351001

Минск 2025

Задание:

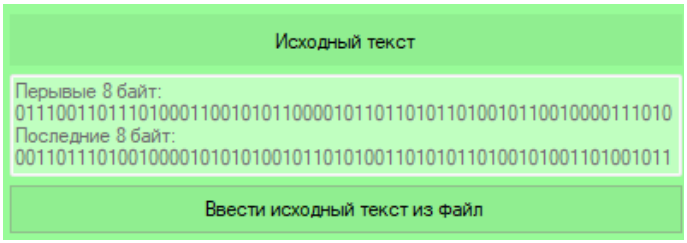
Реализовать систему потокового шифрования и дешифрования для файла с любым содержимым с помощью генератора ключевой последовательности на основе линейного сдвигового регистра с обратной связью LFSR1 (размерность регистра ***36***). Начальное состояние регистра ввести с клавиатуры. Поле для ввода состояния регистра должно игнорировать любые символы кроме 0 и 1. Вывести на экран сгенерированный ключ (последовательность из 0 и 1), исходный файл и зашифрованный файл в двоичном виде. Программа не должна быть написана в консольном режиме. Результат работы программы – зашифрованный/расшифрованный файл.

Примитивный многочлен

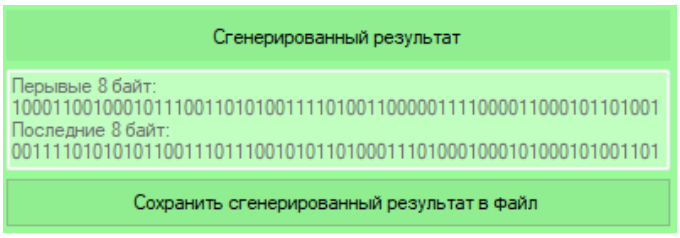
$x^{36} + x^{11} + 1$

Работа с файлами

Взятие исходного файла:



Загрузка результата файла:



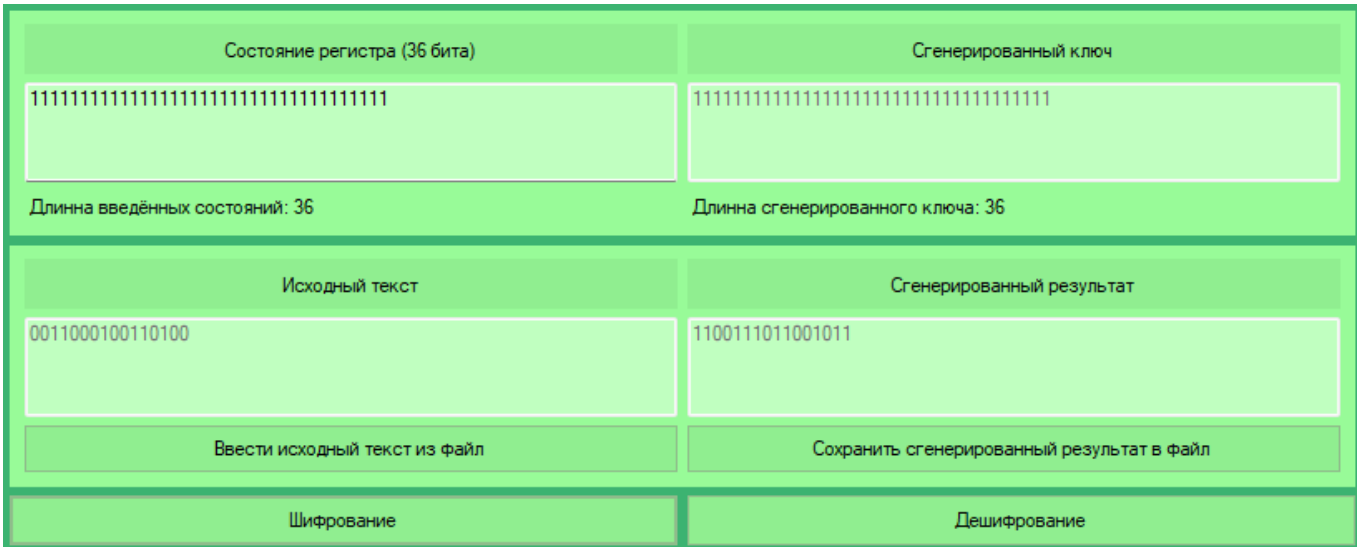
Общий внешний вид страницы:

Тесты

1. Простой ключ, небольшой текст.

Ключ: `111111111111111111111111111111111111`
Исходный текст: `0011000100110100`
Шифротекст: `1100111011001011`

Шифрование:



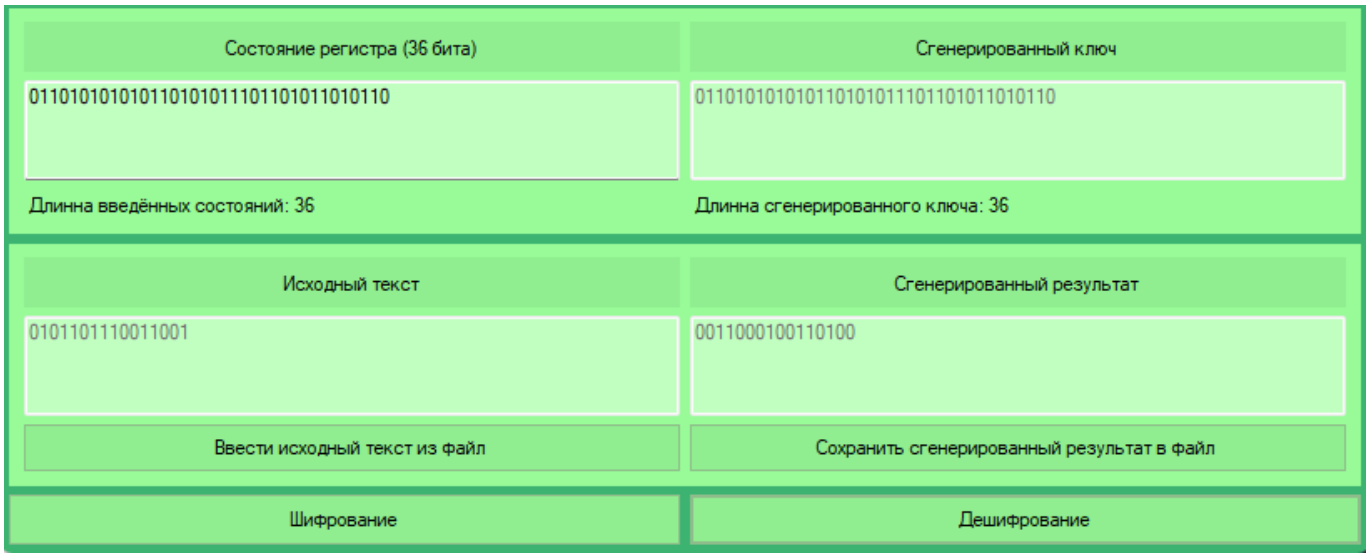
Дешифрирование:

2. Сложный ключ, небольшой текст.

Ключ: `011010101010110101011101101011010110`
Исходный текст: `0011000100110100`
Шифротекст: `0101101110011001`

Шифрование:

Дешифрирование:



3. Простой ключ, большой текст.

Ключ: `111111111111111111111111111111111111`

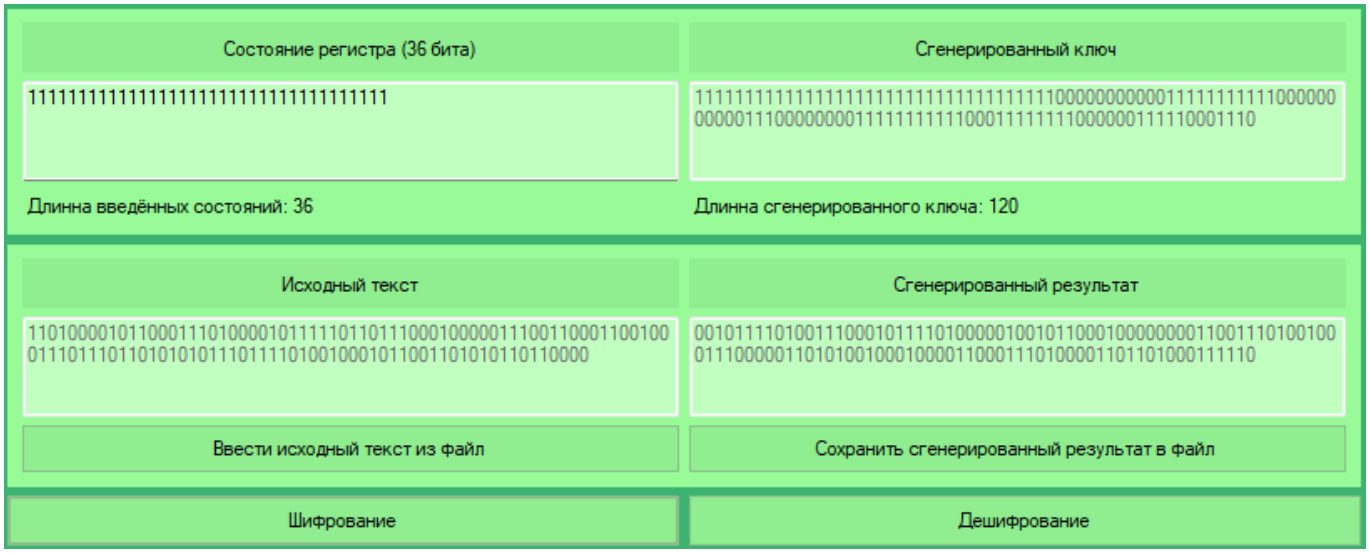
Исходный текст:

```
11010000101100011101000010111110110100001011101111010001100011001101000110001000110100001011111011010000101110010010000
```

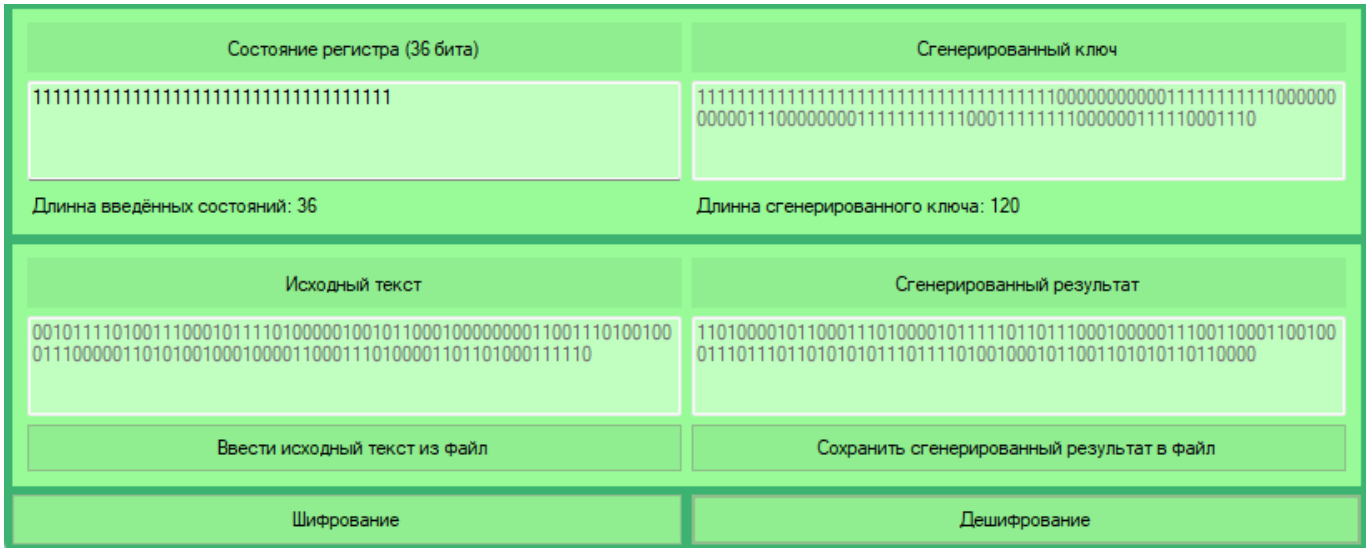
Шифротекст:

```
00101111010011100010111101000001001001011110111010000100110111111110001010111011111000011010111111000001101010011101000
```

Шифрование:



Дешифрирование:



4. Сложный ключ, большой текст.

Ключ: `01101010101011010101110110101101010110`

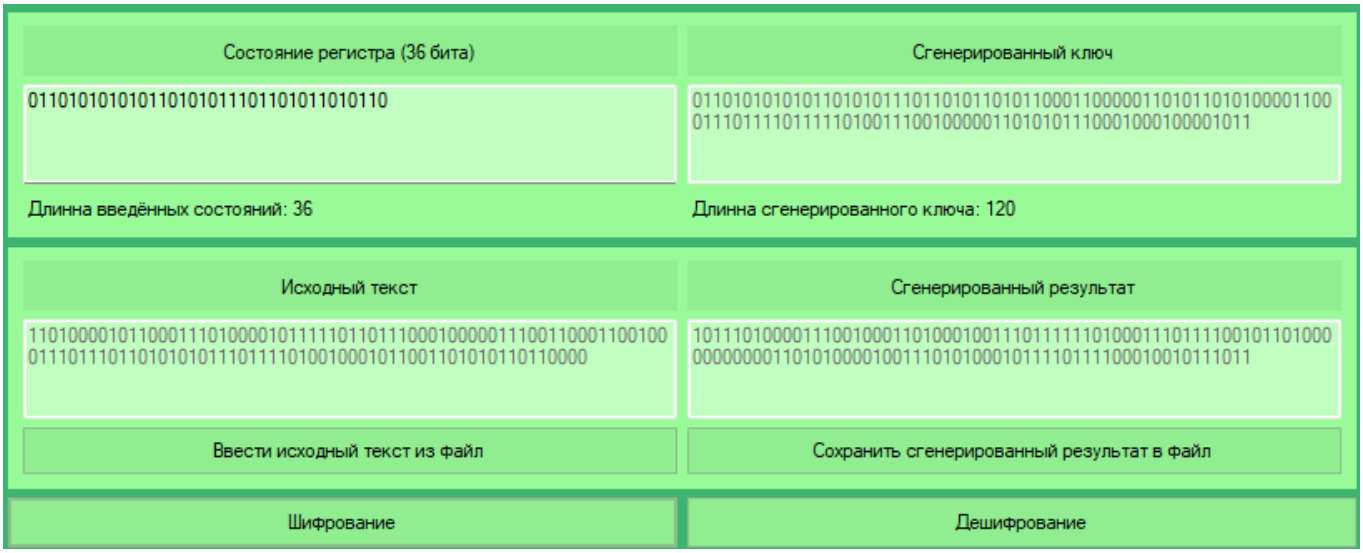
Исходный текст:

```
11010000101100011101000010111110110100001011101111010001100011001101000110001000110100001011111011010000101110010010000
```

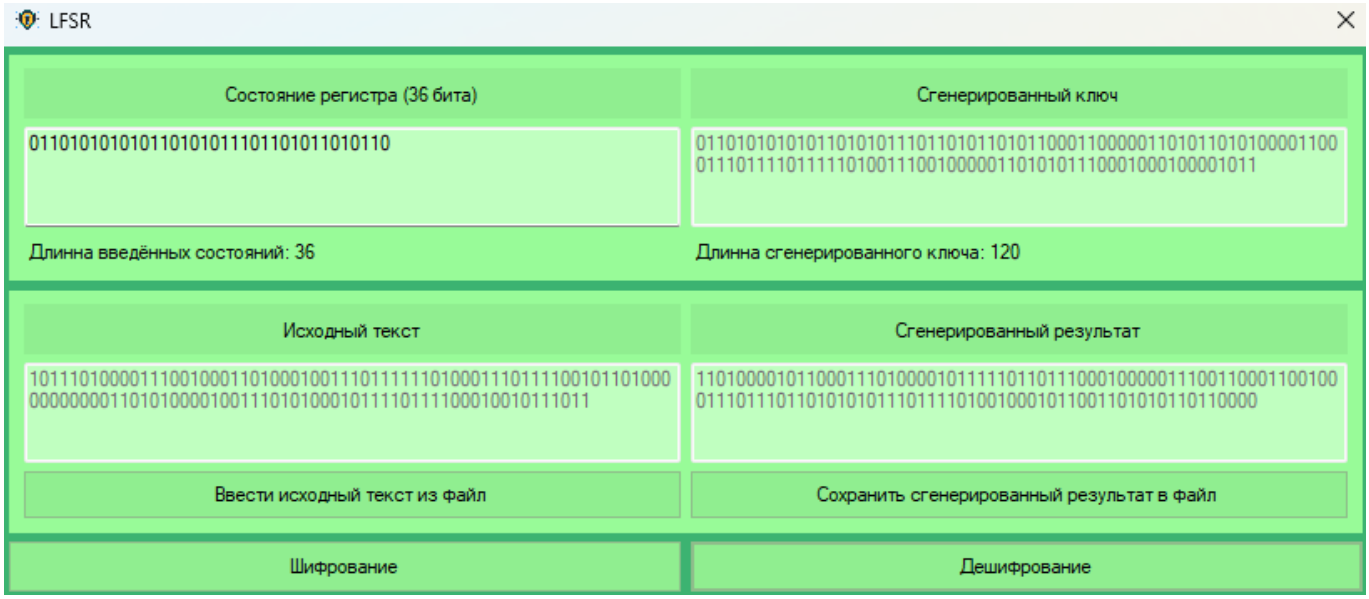
Шифротекст:

```
10111010000111001000110100010011101101000111000010110101010100001001011101010100100110110111111001001101101111111110011110
```

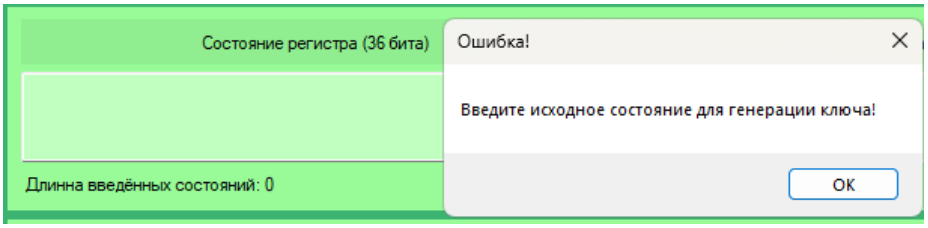
Шифрование:

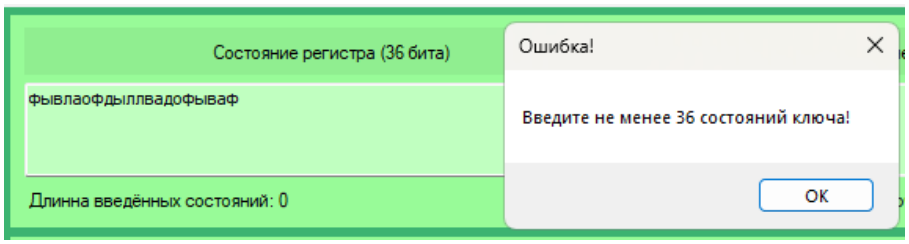


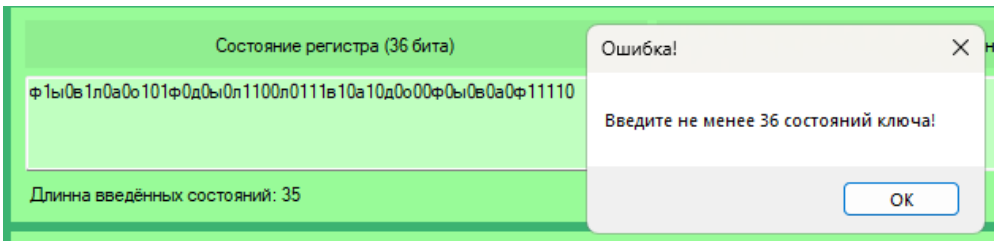
Дешифрирование:

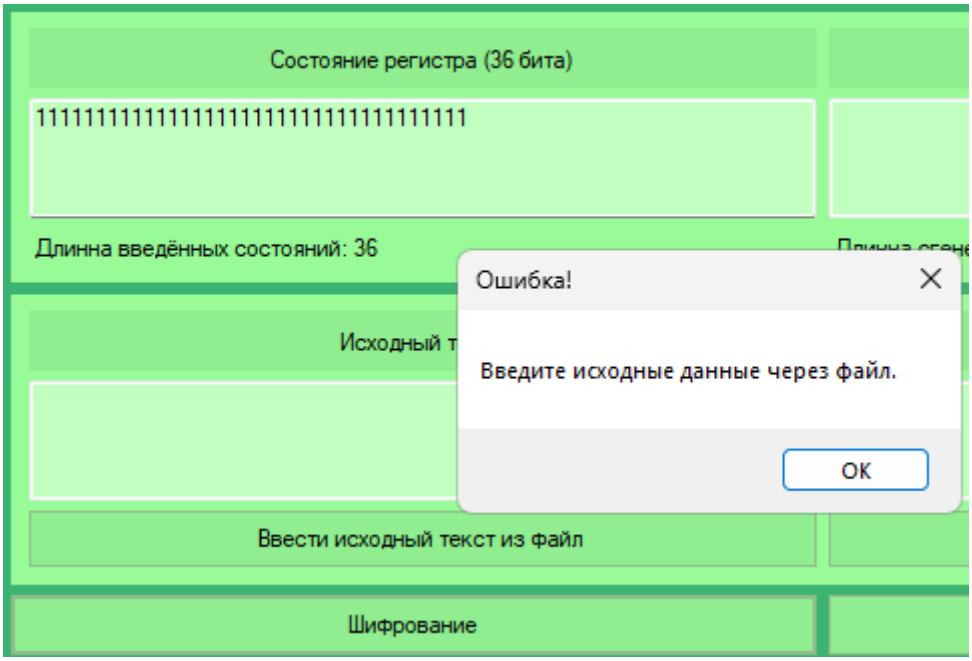


5. Проверка на не валидных данных.









Исходный код

```
using System;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Web;
using System.Windows.Forms;

namespace ToI_2
{
    public partial class MainForm : Form
    {
        private char[] goodKeys = { '0', '1' };

        private string PLAIN_TEXT;
        private string GENERATED_KEY;
        private string CIPER_TEXT;

        public MainForm()
        {
            InitializeComponent();
            setCurrentKeyLength(0);
        }

        private void setCurrentKeyLength(int length)
        {
            keyLengthLabel.Text = $"Длинна введённых состояний: {length}";
        }
```

```
        private string getCurrentKey(string str)
        {
            string res = "";
            for (int i = 0; i < str.Length; i++)
                if (goodKeys.Contains(str[i]))
                    res += str[i];
            return res;
        }
        private (bool, string) getCurrentKeyString(string str)
        {
            if (str.Length == 0)
                return (false, "Введите исходное состояние для генерации ключа!");
            string res = getCurrentKey(str);
            if (res.Length < 36)
                return (false, "Введите не менее 36 состояний ключа!");
            return (true, res);
        }

        private string getFirstBytes(string str, int bytes)
        {
            return str.Substring(0, bytes * 8);
        }

        private string getLastBytes(string str, int bytes)
        {
            return str.Substring(str.Length - bytes * 8, bytes * 8);
        }

        private void setGeneratedText(TextBox textBox, string str)
        {
            const int bytesToSee = 8;
            if (str.Length > bytesToSee * 8 * 2)
                textBox.Text = $"Перывые 8 байт: {getFirstBytes(str,
bytesToSee)}\n\nПоследние 8 байт: {getLastBytes(str, bytesToSee)}";
            else
                textBox.Text = str;
        }

        private char[] shiftKey(char[] key)
        {
            for (int i = 0; i < key.Length - 1; i++)
                key[i] = key[i + 1];
            return key;
        }
        private char xorNums(char bit1, char bit2)
        {
            int num1 = bit1 - '0';
            int num2 = bit2 - '0';
            int result = num1 ^ num2;
            char xorResult = (char)(result + '0');
            return xorResult;
        }
        private char getNewValue(int[] genPos, char[] key)
        {
            char first;
            char second = key[genPos[0] - 1];
            for (var i = 1; i < genPos.Length; i++)
            {
                first = second;
                second = key[genPos[i] - 1];
                second = xorNums(first, second);
            }
```

```
    return second;
}
private string generateByLFSR(string register, int length)
{
    char[] genKey = register.ToArray();
    char[] currentKey = register.ToArray();
    int[] xorPositions = { 36, 11 }; // from 1 .. 36

    StringBuilder bits = new StringBuilder();
    for (var i = 0; i < currentKey.Length; i++)
    {
        bits.Append(currentKey[i]);
    }

    while (bits.Length < length)
    {
        genKey = shiftKey(genKey);
        genKey[genKey.Length - 1] = getNewValue(xorPositions, genKey);
        bits.Append(genKey[genKey.Length - 1]);
    }

    return bits.ToString();
}

private void keyTextBox_TextChanged(object sender, EventArgs e)
{
    setCurrentKeyLength(getCurrentKey(keyTextBox.Text).Length);
}

private void genKeyButton_Click(object sender, EventArgs e)
{
    //string str;
    //bool isCorrect;
    //(isCorrect, str) = getCurrentKeyString(keyTextBox.Text);
    //if (!isCorrect)
    //{
    //    MessageBox.Show(str, "Ошибка!");
    //    return;
    //}
    // generate
    //string key = generateByLFSR(str.Substring(0, 36));
    //setGeneratedKey(key);
}

private string encryptionAlghoritm(string plain, string key)
{
    int test = key.Length;
    StringBuilder bits = new StringBuilder();
    for (int i = 0; i < plain.Length; i++)
    {
        char p = plain[i];
        char k = key[i];
        bits.Append(xorNums(p, k));
    }
    return bits.ToString();
}

private void ENCRIPTION()
{
    /* Key part */
    string key;
    bool isCorrect;
```

```
            (isCorrect, key) = getCurrentKeyString(keyTextBox.Text);
            if (!isCorrect)
            {
                MessageBox.Show(key, "Ошибка!");
                return;
            }

            /* Plain text part */
            if (PLAIN_TEXT == null || PLAIN_TEXT.Length == 0)
            {
                MessageBox.Show("Введите исходные данные через файл.", "Ошибка!");
                return;
            }

            /* Generate key part */
            GENERATED_KEY = generateByLFSR(key.Substring(0, 36), PLAIN_TEXT.Length);
            setGeneratedText(genKeyTextBox, GENERATED_KEY);

            /* Cipher part */
            CIPER_TEXT = encryptionAlghoritm(PLAIN_TEXT, GENERATED_KEY);
            setGeneratedText(cypherTextBox, CIPER_TEXT);

            /* Enables */
            inFileButton.Enabled = true;
            genKeySizeLabel.Text = $"Длинна сгенерированного ключа: {GENERATED_KEY.Length}";
        }
        private void button1_Click(object sender, EventArgs e)
        {
            ENCRIPTION();
        }

        private (bool, string) ReadFileAsBits(string filePath)
        {
            try
            {
                StringBuilder bits = new StringBuilder();

                using (FileStream fs = new FileStream(filePath, FileMode.Open,
FileAccess.Read))
                {
                    int byteRead;
                    while ((byteRead = fs.ReadByte()) != -1)
                    {
                        bits.Append(Convert.ToString(byteRead, 2).PadLeft(8, '0'));
                    }
                }
                if (bits.Length > 0)
                    return (true, bits.ToString());

                return (false, null);
            }
            catch
            {
                return (false, null);
            }
        }

        private (bool, string) WriteFileAsBits(string filePath)
        {
            string bits = CIPER_TEXT;
            try
            {
```

```
                using (FileStream fs = new FileStream(filePath, FileMode.Create,
FileAccess.Write))
                {
                    for (int i = 0; i < bits.Length; i += 8)
                    {
                        string byteBits = bits.Substring(i, Math.Min(8, bits.Length - i));
                        byte byteToWrite = Convert.ToByte(byteBits, 2);
                        fs.WriteByte(byteToWrite);
                    }
                }

                return (true, null);
            }
            catch
            {
                return (false, null);
            }
        }

        private void fromFileButton_Click(object sender, EventArgs e)
        {
            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                bool status;
                string result;
                (status, result) = ReadFileAsBits(openFileDialog.FileName);
                if (status) {
                    PLAIN_TEXT = result;
                    setGeneratedText(plainTextBox, PLAIN_TEXT);
                    return;
                }
                MessageBox.Show("Данные в файле не являются корректными.", "Ошибка!");
            }
        }

        private void inFileButton_Click(object sender, EventArgs e)
        {
            if (saveFileDialog.ShowDialog() == DialogResult.OK)
            {
                string str;
                bool status;
                (status, str) = WriteFileAsBits(saveFileDialog.FileName);
                MessageBox.Show(status ? "Данные успешно записаны в файл." : "Ошибка при
записи в файл.", status ? "Успех" : "Ошибка");

            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            ENCRIPTION();
        }
    }
}
```