БУ ВО «Сургутский государственный университет»

Политехнический институт

Кафедра автоматизированных систем обработки информации и управления

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

ПО ДИСЦИПЛИНЕ «Теория Информации»

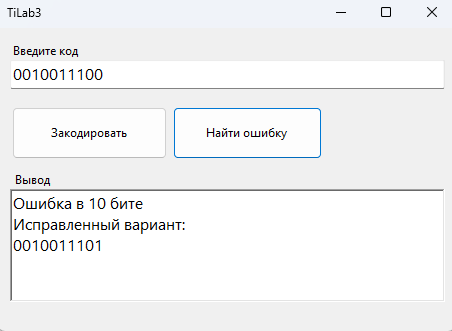
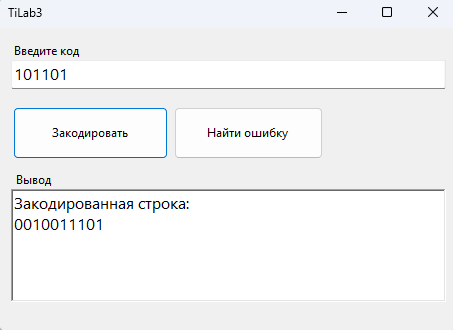
Выполнил: студент группы №606-12,

Речук Дмитрий Максимович

Принял: ст. преподаватель кафедры АСОИУ,

Гавриленко Анна Владимировна

Сургут 2025



using System.Collections;

namespace Hamming

{

*public* *partial* class Form1 : Form

    {

*public* Form1()

        {

            InitializeComponent();

        }

*public* int ERROR\_VAL = -1;

*public* int EMERGENCY = -1;

*public* *static* BitArray Code(string inMessage)

        {

            if (!IsValidBinaryString(inMessage))

*throw* new ArgumentException("Входное сообщение должно содержать только 0 и 1.");

            var messageArray = new BitArray(inMessage.Length, false);

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

                messageArray[i] = inMessage[i] == '1';

            int messageInd = 0;

            int retInd = 0;

            int controlIndex = 1;

            var retArray = new BitArray(messageArray.Length + 1 + (int)Math.Ceiling(Math.Log(messageArray.Length, 2)));

            while (messageInd < messageArray.Length)

            {

                if (retInd + 1 == controlIndex)

                {

                    retInd++;

                    controlIndex \*= 2;

*continue*;

                }

                retArray.Set(retInd, messageArray.Get(messageInd));

                messageInd++;

                retInd++;

            }

            retInd = 0;

            controlIndex = 1 << (int)Math.Log(retArray.Length, 2);

            while (controlIndex > 0)

            {

                int c = controlIndex - 1;

                int counter = 0;

                while (c < retArray.Length)

                {

                    for (int i = 0; i < controlIndex && c < retArray.Length; i++)

                    {

                        if (retArray.Get(c))

                            counter++;

                        c++;

                    }

                    c += controlIndex;

                }

                if (counter % 2 != 0) retArray.Set(controlIndex - 1, true);

                controlIndex /= 2;

            }

*return* retArray;

        }

*public* BitArray Decode(string inMessage)

        {

            if (!IsValidBinaryString(inMessage))

*throw* new ArgumentException("Входное сообщение должно содержать только 0 и 1.");

            var codedArray = new BitArray(inMessage.Length, false);

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

                codedArray[i] = inMessage[i] == '1';

            var decodedArray = new BitArray((int)(codedArray.Count - Math.Ceiling(Math.Log(codedArray.Count, 2))), false);

            int count = 0;

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

            {

                for (int j = 0; j < Math.Ceiling(Math.Log(codedArray.Count, 2)); j++)

                {

                    if (i == Math.Pow(2, j) - 1)

                        i++;

                }

                decodedArray[count++] = codedArray[i];

            }

            string strDecodedArray = "";

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

                strDecodedArray += decodedArray[i] ? "1" : "0";

            var checkArray = Code(strDecodedArray);

            byte[] failBits = new byte[checkArray.Length - decodedArray.Length];

            count = 0;

            bool isMistake = false;

            for (int i = 0; i < checkArray.Length - decodedArray.Length; i++)

            {

                if (codedArray[(int)Math.Pow(2, i) - 1] != checkArray[(int)Math.Pow(2, i) - 1])

                {

                    failBits[count++] = (byte)(Math.Pow(2, i));

                    isMistake = true;

                }

            }

            if (isMistake)

            {

                int mistakeIndex = 0;

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

                    mistakeIndex += failBits[i];

                mistakeIndex--;

                if (mistakeIndex >= 0 && mistakeIndex < codedArray.Length)

                {

                    codedArray.Set(mistakeIndex, !codedArray[mistakeIndex]);

                    ERROR\_VAL = mistakeIndex;

                }

                else

                {

*throw* new InvalidOperationException("Ошибка за пределами диапазона закодированного сообщения.");

                }

                count = 0;

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

                {

                    for (int j = 0; j < Math.Ceiling(Math.Log(codedArray.Count, 2)); j++)

                    {

                        if (i == Math.Pow(2, j) - 1)

                            i++;

                    }

                    decodedArray[count++] = codedArray[i];

                }

            }

*return* decodedArray;

        }

*private* void Button1\_Click(object sender, EventArgs e)

        {

            try

            {

                richTextBox1.Clear();

                BitArray code = Decode(textBox1.Text);

                if (ERROR\_VAL == -1)

                {

                    MessageBox.Show("В коде ошибки нет.");

*return*;

                }

                richTextBox1.AppendText($"Ошибка в {ERROR\_VAL + 1} бите\nИсправленный вариант:\n");

                char[] textArray = textBox1.Text.ToCharArray();

                textArray[ERROR\_VAL] = textArray[ERROR\_VAL] == '0' ? '1' : '0';

                richTextBox1.AppendText(new string(textArray));

            }

            catch (ArgumentException ex)

            {

                MessageBox.Show($"Ошибка ввода: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Непредвиденная ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

*private* void Button2\_Click(object sender, EventArgs e)

        {

            try

            {

                richTextBox1.Clear();

                if (string.IsNullOrWhiteSpace(textBox1.Text))

                {

                    MessageBox.Show("Введите сообщение для кодирования.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

*return*;

                }

                richTextBox1.AppendText("Закодированная строка:\n");

                BitArray code = Code(textBox1.Text);

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

                    richTextBox1.AppendText(code[i] ? "1" : "0");

            }

            catch (ArgumentException ex)

            {

                MessageBox.Show($"Ошибка ввода: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Непредвиденная ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

*private* *static* bool IsValidBinaryString(string input)

        {

            foreach (char c in input)

                if (c != '0' && c != '1')

*return* false;

*return* true;

        }

    }

}