Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.01 – «Информатика и вычислительная техника»

Дисциплина: «Защита информации»

Профиль: «Разработка программно-информационных систем»

Семестр 5

ОТЧЕТ

по лабораторной работе №5

Тема: «Шифры обнаружения и коррекции ошибок»

Выполнил: студент группы РИС 20-2б

Уржумов В.И. \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Шереметьев В. Г. \_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_

Пермь, 2022

**ЦЕЛЬ РАБОТЫ**

Получить практические навыки по применению шифров обнаружения и коррекции ошибок.

**ЗАДАНИЕ**

Реализовать проверку правильности передачи двоичной кодовой последовательности длиной 8 бит, используя метод кода с одиночным битом четности.

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**Обнаружение ошибок.**

Каналы передачи данных ненадежны, да и само оборудование обработки информации работает со сбоями. По этой причине важную роль приобретают механизмы детектирования ошибок. Ведь если ошибка обнаружена, можно осуществить повторную передачу данных и решить проблему. Если исходный код по своей длине равен полученному коду, обнаружить ошибку передачи не предоставляется возможным.

Простейшим способом обнаружения ошибок является контроль по четности. Обычно контролируется передача блока данных (М бит). Этому блоку ставится в соответствие кодовое слово длиной N бит, причем N>M. Избыточность кода характеризуется величиной 1-M/N. Вероятность обнаружения ошибки определяется отношением M/N (чем меньше это отношение, тем выше вероятность обнаружения ошибки, но и выше избыточность).

При передаче информации она кодируется таким образом, чтобы с одной стороны характеризовать ее минимальным числом символов, а с другой – минимизировать вероятность ошибки при декодировании получателем. Для выбора типа кодирования важную роль играет так называемое расстояние Хэмминга.

Пусть А и Б две двоичные кодовые последовательности равной длины. Расстояние Хэмминга между двумя этими кодовыми последовательностями равно числу символов, которыми они отличаются. Например, расстояние Хэмминга между кодами 00111 и 10101 равно 2.

Можно показать, что для детектирования ошибок в n битах, схема кодирования требует применения кодовых слов с расстоянием Хэмминга не менее N+1. Можно также показать, что для исправления ошибок в N битах необходима схема кодирования с расстоянием Хэмминга между кодами не менее 2N+1. Таким образом, конструируя код, мы пытаемся обеспечить расстояние Хэмминга между возможными кодовыми последовательностями больше, чем оно может возникнуть из-за ошибок.

Широко распространены коды с одиночным битом четности. В этих кодах к каждым М бит добавляется 1 бит, значение которого определяется четностью (или нечетностью) суммы этих М бит. Так, например, для двухбитовых кодов 00, 01, 10, 11 кодами с контролем четности будут 000, 011, 101 и 110. Если в процессе передачи один бит будет передан неверно, четность кода из М+1 бита изменится.

Предположим, что частота ошибок (BER) равна р=10-4. В этом случае вероятность передачи 8 бит с ошибкой составит 1-(1-p)8=7,9х10-4. Добавление бита четности позволяет детектировать любую ошибку в одном из переданных битах. Здесь вероятность ошибки в одном из 9 бит равна 9p(1-p)8. Вероятность же реализации необнаруженной ошибки составит 1-(1-p)9 – 9p(1-p)8 = 3,6x10-7. Таким образом, добавление бита четности уменьшает вероятность необнаруженной ошибки почти в 1000 раз. Использование одного бита четности типично для асинхронного метода передачи. В синхронных каналах чаще используется вычисление и передача битов четности как для строк, так и для столбцов передаваемого массива данных. Такая схема позволяет не только регистрировать но и исправлять ошибки в одном из битов переданного блока.

Контроль по четности достаточно эффективен для выявления одиночных и множественных ошибок в условиях, когда они являются независимыми. При возникновении ошибок в кластерах бит метод контроля четности неэффективен и тогда предпочтительнее метод вычисления циклических сумм (CRC). В этом методе передаваемый кадр делится на специально подобранный образующий полином. Дополнение остатка от деления и является контрольной суммой.

**ХОД РАБОТЫ**

На рисунке 1 представлена главная форма программы. Пример работы программы представлен на рисунках 2,3.

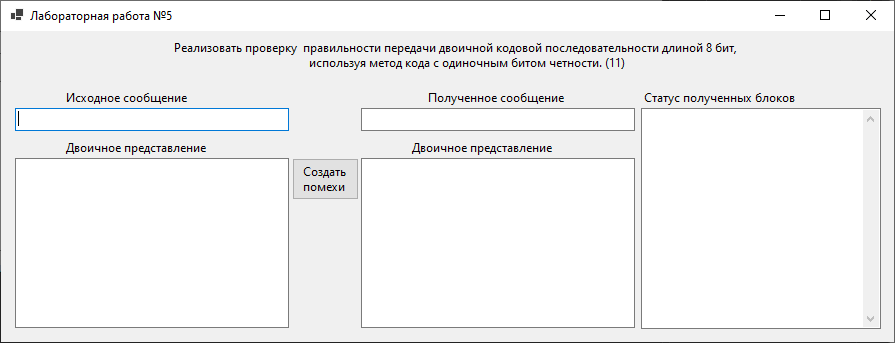


Рисунок 1- Главная форма

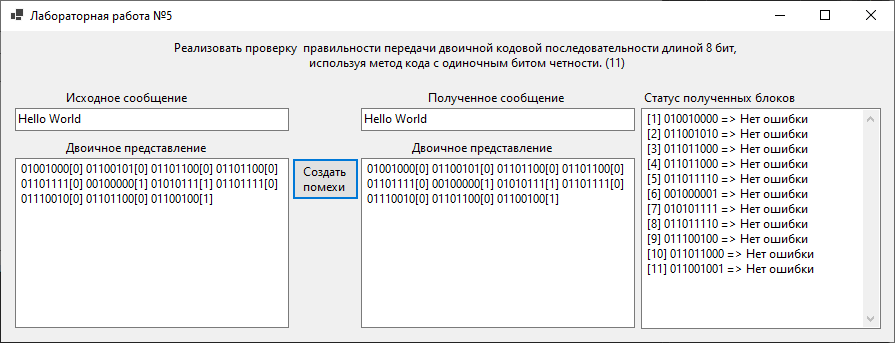


Рисунок 2-Пример работы программы

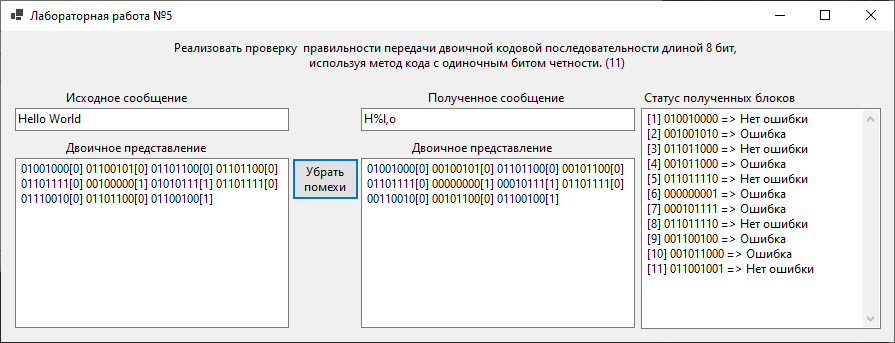


Рисунок 3- Пример работы программы

**ПРИЛОЖЕНИЕ А**

**Листинг класса Labs5**

public partial class Labs5 : Form

{

string[,] errors;

bool interferences = false;

public Labs5()

{

InitializeComponent();

}

private void Labs5\_FormClosed(object sender, FormClosedEventArgs e)

{

MainForm mf = new MainForm();

mf.Show();

}

private string CheckString(char c)

{

string res = Convert.ToString(c, 2);

while (res.Length < 8)

res = "0" + res;

return res;

}

private void ContinuousTransmission()

{

BinaryInputMessage.Text = "";

foreach (var v in InputMessage.Text)

BinaryInputMessage.Text += CheckString(v) + $"[{AddControlBit(Convert.ToString(v, 2))}]" + " ";

BinaryOutputMessage.Text = Decrypt(BinaryInputMessage.Text);

ListErrors();

}

private void InputMessage\_TextChanged(object sender, EventArgs e)

{

if ((!Regex.IsMatch(InputMessage.Text, @"^[a-zA-Z0-9 ]+$"))&&InputMessage.Text.Length>0)

MessageBox.Show("Доступны только 0-9 и a-z(A-Z)");

else

ContinuousTransmission();

}

private string AddControlBit(string s)

{

int[] mass = s.Select(ch => int.Parse(ch.ToString())).ToArray();

int[] mass1 = new int[mass.Length];

mass1 = mass;

int count = 0;

string result = "";

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

count += mass[i] \* mass1[i];

if (count > 1)

result = Convert.ToString(count % 2);

else

result = Convert.ToString(count);

return result;

}

private string ControlBitDecoding(string str)

{

int[] mass = str.Select(ch => int.Parse(ch.ToString())).ToArray();

int a = Convert.ToInt16(mass[mass.Length - 1]);

int[] mass1 = new int[mass.Length];

mass1 = mass;

int count = 0;

int result;

for (int i = 0; i < mass.Length - 1; i++)

count += mass[i] \* mass1[i];

if (count > 1)

result = count % 2;

else

result = count;

if (result == a)

return "Нет ошибки";

else

return "Ошибка";

}

private string Interferences(string input)

{

string temp = "";

Random rand = new Random();

int error = rand.Next(0, 2);

foreach (var v in input)

{

if (v != ' ')

{

if (v == '1' && error == 1)

{

temp += 0;

error = 0;

}

else

temp += v;

}

else

{

temp += " ";

error = rand.Next(0, 2);

}

}

return temp;

}

private string Decrypt(string encryptedText)

{

if (interferences == true)

encryptedText = Interferences(encryptedText);

int count = 1;

foreach (var v in encryptedText)

if (v == ' ')

count++;

errors = new string[count - 1, 2];

string result = "";

string temp = "";

int i = 0;

foreach (var v in encryptedText)

{

if (v != ' ')

{

if (v != '[' && v != ']')

temp += v;

}

else

{

var strArr = Enumerable.Range(0, temp.Length / 8).Select(i => Convert.ToByte(temp.Substring(i \* 8, 8), 2)).ToArray();

result += Encoding.UTF8.GetString(strArr);

errors[i, 0] = temp;

errors[i, 1] = ControlBitDecoding(temp);

i++;

temp = "";

}

}

OutputMessage.Text = result;

return encryptedText;

}

private void ListErrors()

{

ErrorArray.Text = "";

for (int i = 0; i < errors.Length / 2; i++)

ErrorArray.Text += $"[{i + 1}] {errors[i, 0]} => {errors[i, 1]}{Environment.NewLine}";

}

private void button2\_Click(object sender, EventArgs e)

{

if (button2.Text == "Создать помехи\r\n")

{

interferences = true;

button2.Text = "Убрать помехи";

}

else

{

interferences = false;

button2.Text = "Создать помехи\r\n";

}

ContinuousTransmission();

}

}