**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«Сибирский государственный университет науки и технологий   
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информационно-управляющих систем

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Теория информации

|  |
| --- |
| Линейные групповые коды |

Руководитель А.Н. Бочаров

подпись, дата инициалы, фамилия

Обучающийся БПИ22-02, 221217039 Н.Ю. Сюткин

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2024 г.

# ЦЕЛЬ РАБОТЫ

Закрепление знаний по методам кодирования информации.

# порядок выполнения работы

1. Ознакомиться с общей̆ постановкой̆ задачи.
2. Ознакомится с вариантом задания – соответствует вашему номеру в списке группы.
3. Выполнить задания согласно варианту.
4. Подготовить отчет по лабораторной̆ работе. Отчет должен включать в себя:   
   •  титульный̆ лист;   
   •  цель лабораторной̆ работы;   
   •  постановку задачи;   
   •  ход работы;   
   •  краткие ответы на контрольные вопросы;   
   •  выводы по лабораторной̆ работе.
5. Защитить лабораторную работу перед преподавателем.

# постановка задачи

1. Построить линейный групповой код, способный исправлять одиночную ошибку. Вариант взять аналогичный из лабораторной работы № 4.   
 2. Привести пример построения 10 кодовых комбинаций.   
 3. Показать процесс исправления ошибки в заданном разряде.

4. Составить программу, кодирующую и декодирующую кодовую комбинацию, с целью обнаружения и исправления одиночной ошибки.

Вариант 21. Кол-во сообщений (n) - 16.

# ХОД РАБОТЫ

Кол-во сообщений(n) = 16 = 24 => nи = 4

nк = = ≈ 3 бит

n = nк + nи = 3 + 4 = 7

Wп ≥ dmin – 1 => Wп ≥ 3 – 1 => Wп ≥ 2

G4, 3

1. 0000 5) 0100 9) 1000 13) 1100
2. 0001 6) 0101 10) 1001 14) 1101
3. 0010 7) 0110 11) 1010 15) 1110
4. 0011 8) 0111 12) 1011 16) 1111

Возьмем следующую комбинацию.

|  |  |  |  |
| --- | --- | --- | --- |
| a1 | a2 | a3 | a4 |
| 1 | 0 | 1 | 0 |

Единицы в данной комбинации находятся в 1,3 позициях. Поэтому суммируем по модулю два 1,3 строки матрицы П

111 + 101= 010 => p1=0 p2=1 p3= 0

Таким образом, итоговый код будет выглядеть следующим образом.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | p1 | p2 | p3 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |

H = ПT \* Inк =

S1 = p1 + a1 + a2 + a3

S2 = p2 + a1 + a2 + a4

S3 = p3 + a1 + a3 + a4

Пусть ошибка в 7-м разряде. Тогда код будет выглядеть следующим образом.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | p1 | p2 | p3 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 |

S1 = p1 + a1 + a2 + a3 = 0 + 1 + 0 + 1 = 0

S2 = p2 + a1 + a2 + a4 = 1 + 1 + 0 + 0 = 0

S3 = p3 + a1 + a3 + a4 = 1 + 1 + 1 + 0 = 1

S

Это соответствует 7-му столбцу матрицы H => ошибка в 7-м разряде.

Программа, реализующая кодирование/декодирование линейных групповых кодов.

Демонстрация работы программы (Рисунок 1).

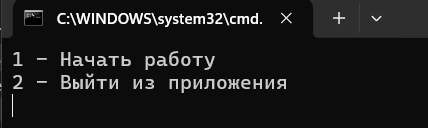


Рисунок 1 – Меню программы

Нажимая действие 1, генерируем 10 кодов (Рисунок 2).

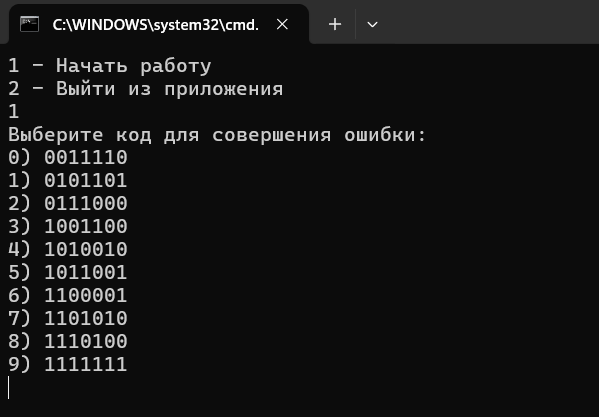


Рисунок 2 – Кодирование

Далее выбираем разряд для ошибки, и программа обнаружит и исправит ошибку в заданном разряде методом декодирования ЛГК (Рисунок 3).

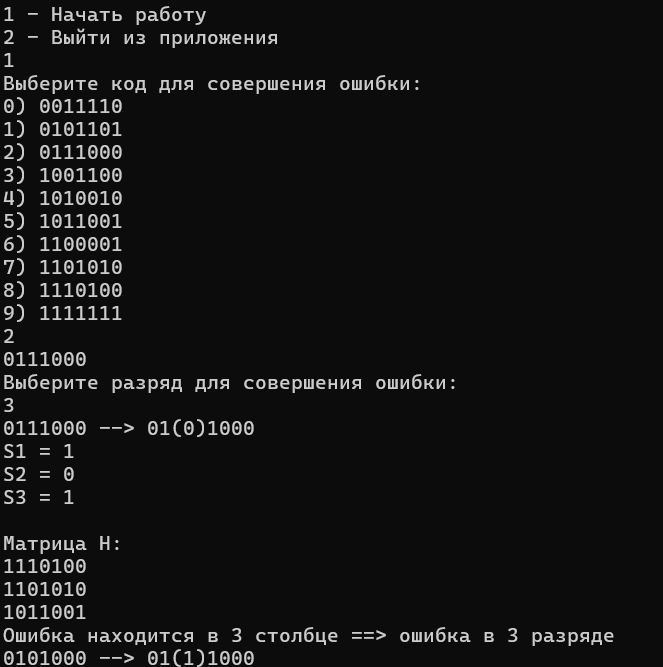


Рисунок 3 – Декодирование

Действие 2 прекращает работу программы (Рисунок 4):

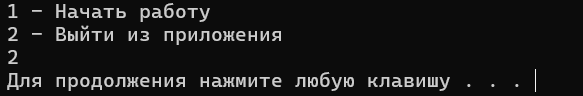


Рисунок 4 – Прекращение работы программы

Листинг программы:

using System;

using System.Collections.Generic;

using System.IO.Pipes;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

namespace lab5

{

internal class Class1

{

public static void ShowMatrix(int[,] Matrix)

{

for (int i = 0; i < Matrix.GetLength(0); i++)

{

for (int j = 0; j < Matrix.GetLength(1); j++)

{

Console.Write(Matrix[i, j]);

}

Console.WriteLine();

}

}

public static int[] GetCurrComb(int[,] Matrix, int idx)

{

int[] curr\_comb = new int[Matrix.GetLength(1)];

for (int j = 0; j < Matrix.GetLength(1); j++)

{

curr\_comb[j] = Matrix[idx, j];

}

return curr\_comb;

}

public static int[,] DeleteOneComb(int[,] Matrix, int idx)

{

int count = -1;

int[,] Result\_Matrix = new int[Matrix.GetLength(0) - 1, Matrix.GetLength(1)];

for (int i = 0; i < Matrix.GetLength(0); i++)

{

if (i == idx)

{

continue;

}

else

{

count++;

for (int j = 0; j < Matrix.GetLength(1); j++)

{

Result\_Matrix[count, j] = Matrix[i, j];

}

}

}

return Result\_Matrix;

}

public static int[] SummModule2(int[] x, int[] y)

{

int[] result = new int[x.Length];

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

{

if (x[i] == y[i])

result[i] = 0;

else

result[i] = 1;

}

return result;

}

public static int[] LinearCodingOnce(int[,] Matrix, int[] comb)

{

List<int> indexes = new List<int>();

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

{

if (comb[i] == 1)

indexes.Add(i);

}

int[][] Temp = new int[indexes.Count][];

for (int i = 0; i < indexes.Count; i++)

{

Temp[i] = GetCurrComb(Matrix, indexes[i]);

}

int[] result = new int[comb.Length];

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

{

if (i == 0)

result = SummModule2(Temp[i], Temp[i + 1]);

else

result = SummModule2(result, Temp[i + 1]);

}

int[] FinishCode = new int[comb.Length + result.Length];

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

{

FinishCode[i] = comb[i];

}

int count = 0;

for(int i = comb.Length; i < comb.Length + result.Length; i++)

{

FinishCode[i] = result[count++];

}

return FinishCode;

}

public static int[][] LinearCoding(int[,] Matrix)

{

int n\_u = 4;

int n\_k = 3;

int n = n\_u + n\_k;

int[,] Matrix\_All = new int[,]

{

{0, 0, 1, 1 },

{0, 1, 0, 1 },

{0, 1, 1, 0 },

{0, 1, 1, 1 },

{1, 0, 0, 1 },

{1, 0, 1, 0 },

{1, 0, 1, 1 },

{1, 1, 0, 0 },

{1, 1, 0, 1 },

{1, 1, 1, 0 },

{1, 1, 1, 1 }

};

Random rnd = new Random();

int random = rnd.Next(0, Matrix\_All.GetLength(0));

Matrix\_All = DeleteOneComb(Matrix\_All, random);

int[][] Matrix\_Codes = new int[Matrix\_All.GetLength(0)][];

for(int i = 0; i < Matrix\_All.GetLength(0); i++)

{

Matrix\_Codes[i] = LinearCodingOnce(Matrix, GetCurrComb(Matrix\_All, i));

}

return Matrix\_Codes;

}

public static int[] FixMistake(int[] comb, int[,] Matrix\_p\_tr\_)

{

int s1 = (comb[4] + comb[0] + comb[1] + comb[2]) % 2;

int s2 = (comb[5] + comb[0] + comb[1] + comb[3]) % 2;

int s3 = (comb[6] + comb[0] + comb[2] + comb[3]) % 2;

Console.WriteLine("\nS1 = " + s1);

Console.WriteLine("S2 = " + s2);

Console.WriteLine("S3 = " + s3);

int[] ski = new int[3];

ski[0] = s1;

ski[1] = s2;

ski[2] = s3;

Console.WriteLine("\nМатрица H: ");

ShowMatrix(Matrix\_p\_tr\_);

int res\_idx = 0;

for(int j = 0; j < Matrix\_p\_tr\_.GetLength(1); j++)

{

bool isMatch = true;

for(int i = 0; i < Matrix\_p\_tr\_.GetLength(0); i++)

{

if (Matrix\_p\_tr\_[i,j] != ski[i])

{

isMatch = false;

break;

}

}

if(isMatch)

{

res\_idx = j;

Console.WriteLine("Ошибка находится в " + (j + 1) + " столбце ==> ошибка в " + (j + 1) + " разряде");

}

}

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

Console.Write(comb[i]);

if (comb[res\_idx] == 0)

comb[res\_idx] = 1;

else

comb[res\_idx] = 0;

Console.Write(" --> ");

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

{

if (i == res\_idx)

Console.Write("(" + comb[i] + ")");

else

Console.Write(comb[i]);

}

Console.WriteLine();

return comb;

}

public static void Main()

{

int ans = 0;

do

{

Console.WriteLine("1 - Начать работу" +

"\n2 - Выйти из приложения");

ans = Convert.ToInt32(Console.ReadLine());

switch(ans)

{

case 1:

int[,] Matrix\_P = new int[,]

{

{1, 1, 1 },

{1, 1, 0 },

{1, 0, 1 },

{0, 1, 1 }

};

int[,] Matrix\_P\_Tr = new int[,]

{

{1, 1, 1, 0, 1, 0, 0 },

{1, 1, 0, 1, 0, 1, 0 },

{1, 0, 1, 1, 0, 0, 1 }

};

int[][] LinearCodes = LinearCoding(Matrix\_P);

Console.WriteLine("Выберите код для совершения ошибки: ");

for (int i = 0; i < LinearCodes.GetLength(0); i++)

{

for (int j = 0; j < 7; j++)

{

if (j == 0)

Console.Write(i + ") ");

Console.Write(LinearCodes[i][j]);

}

Console.WriteLine();

}

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

int[] chosen\_code = LinearCodes[idx];

for (int i = 0; i < LinearCodes[idx].Length; i++)

Console.Write(LinearCodes[idx][i]);

Console.WriteLine("\nВыберите разряд для совершения ошибки: ");

idx = Convert.ToInt32(Console.ReadLine());

for (int i = 0; i < chosen\_code.Length; i++)

Console.Write(chosen\_code[i]);

if (chosen\_code[idx - 1] == 0)

chosen\_code[idx - 1] = 1;

else

chosen\_code[idx - 1] = 0;

Console.Write(" --> ");

for (int i = 0; i < chosen\_code.Length; i++)

{

if (i == idx - 1)

Console.Write("(" + chosen\_code[i] + ")");

else

Console.Write(chosen\_code[i]);

}

chosen\_code = FixMistake(chosen\_code, Matrix\_P\_Tr);

break;

}

} while (ans != 2);

}

}

}

# ВЫВОДЫ

Таким образом, были получены навыки построения линейных групповых кодов, обнаружения и исправления одной в ошибки в коде.