Лабораторная работа № 3

по ТИПиС

студента группы ИТ-32

Манукова Давида Альбертовича

Выполнение: \_\_\_\_\_\_\_\_\_\_ Защита: \_\_\_\_\_\_\_\_\_\_

**Помехоустойчивое кодирование сообщений**

Задание к работе

***Задание А.***

А.1. Для заданного в работе №2 варианта системы передачи информации определить:

А.1.1. Количество информационных разрядов в кодовом слове,

А.1.2. Минимальное кодовое расстояние для однократных и двукратных ошибок,

А.1.3. Количество контрольных разрядов для кодов, исправляющих однократные и двукратные ошибки.

А.2. Составить порождающие матрицы для кода Хемминга, исправляющего однократные ошибки, и канонического матричного кода, исправляющего двукратные ошибки.

А.3. Составить проверочные матрицы для матричных кодов, исправляющих однократные и двукратные ошибки.

А.4. Для матричного кода, исправляющего двукратные ошибки, составить таблицу соответствия синдрома и локализации ошибок (в таблицу включить только ошибки в информационных разрядах).

А.5. С помощью программы «Калькулятор матричного кодирования» выполнить помехоустойчивое кодирование первичных кодов, составленных в п. А.11 работы №1. При кодировании предполагать однократную ошибку.

А.6. Внести однократные и двукратные ошибки в несколько вторичных кодов, полученных в п. А.5. С помощью программы «Калькулятор матричного кодирования» выполнить декодирование и локализацию внесенных ошибок. Сравнить с правильными первичными кодами.

А.7. Составить порождающие полиномы **g(x)** для исправления однократных и двукратных ошибок.

А.8. С помощью программы «Калькулятор циклических кодов» выполнить помехоустойчивое кодирование первичных кодов, составленных в п. А.11 работы №1. При кодировании предполагать однократную ошибку.

А.9. Внести однократные и двукратные ошибки в несколько вторичных кодов, полученных в п. А.8. С помощью программы «Калькулятор циклических кодов» выполнить декодирование и локализацию внесенных ошибок. Сравнить с правильными первичными кодами.

***Задание Б.***

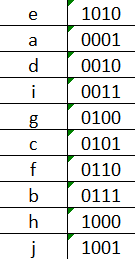
Б.1. Составить программу кодирования, декодирования и локализации однократных ошибок с использованием матричных или циклических кодов (на выбор).

Б.2. Выполнить пп. А.5, А.6 или А.8, А.9 с помощью разработанной программы.

Ход работы

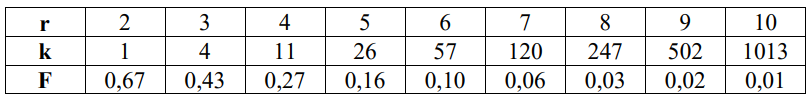
Задание А

1. Количество информационных битов: k = 4.



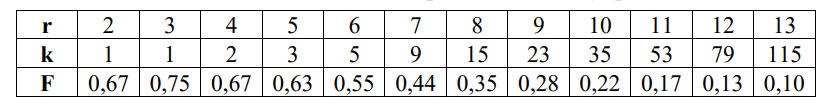
1. Минимальное кодовое расстояние:

* для 1-кратной ошибки: d1 = 2 \* 1 + 1 = 3;
* для 2-кратной ошибки: d2 = 2 \* 2 + 1 = 5.

1. По условию Хемминга: 

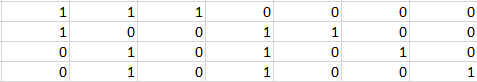
* для исправления 1-кратной ошибки: 2­r – r – 1 ≥ k.

n = r + k = 7, r = 3, k = 4 => Код Хемминга – (7, 4).

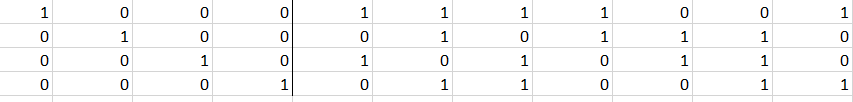


* для исправления 2-кратной ошибки: 2r ≥ => 2r ≥ 64 => r ≥ 6, n = k + r = 10 => Код Хемминга – (10, 4).

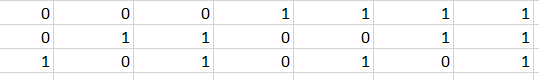
1. Порождающая матрица для кода Хемминга, исправляющего однократные ошибки:



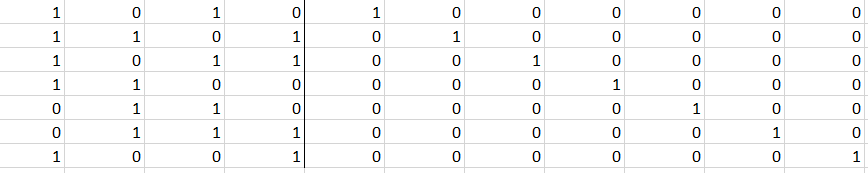
Порождающая матрица для канонического матричного кода, исправляющего двукратные ошибки:



1. Проверочная матрица для кода Хемминга:

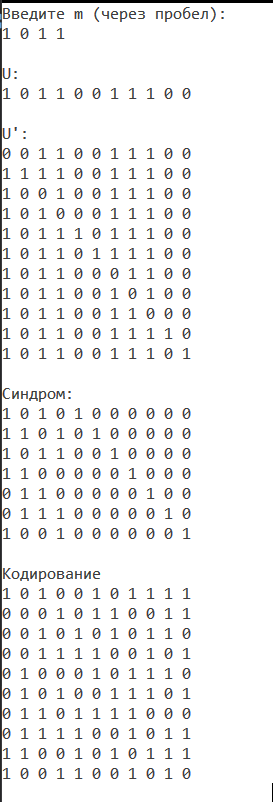


Проверочная матрица для канонического матричного кода:



Задание Б

1. Результат работы программы



1. Листинг программы

class Program

{

static void Main(string[] args)

{

int[,] G\_matrix = new int[4, 11] {{1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1},

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

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

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

int[,] H\_matrix = new int[7, 11] {{1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0},

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

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

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

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

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

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

int[] m = new int[4];

int[] U = new int[11];

int[,] errors = new int[11, 11] {{1,0,0,0,0,0,0,0,0,0,0},

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

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

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

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

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

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

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

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

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

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

int[,] code\_U = new int[11, 11];

int[,] syndromes = new int[7, 11];

int[,] bincodes = {

{1, 0, 1, 0 },

{0, 0, 0, 1 },

{0, 0, 1, 0 },

{0, 0, 1, 1 },

{0, 1, 0, 0 },

{0, 1, 0, 1 },

{0, 1, 1, 0 },

{0, 1, 1, 1 },

{1, 1, 0, 0 },

{1, 0, 0, 1 }

};

int[,] code\_Matrix = new int[10, 11];

Console.WriteLine("Введите m (через пробел):");

string[] str = Console.ReadLine().Split(" ");

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

{

m[i] = Int32.Parse(str[i]);

}

int sum;

Console.WriteLine("\nU:");

for (int i = 0; i < 11; i++)

{

sum = 0;

for (int j = 0; j < 4; j++)

{

sum += (G\_matrix[j, i] \* m[j]);

}

U[i] = sum % 2;

}

for (int i = 0; i < 11; i++)

{

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

}

Console.WriteLine("\n\nU':");

for (int i = 0; i < 11; i++)

{

sum = 0;

for (int j = 0; j < 11; j++)

{

code\_U[j,i] = (U[j] + errors[i, j]) % 2;

}

}

for (int i = 0; i < 11; i++)

{

for (int j = 0; j < 11; j++)

{

Console.Write(code\_U[j,i] + " ");

}

Console.WriteLine();

}

Console.WriteLine("\nСиндром:");

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

{

for (int j = 0; j < 11; j++)

{

sum = 0;

for (int k = 0; k < 11; k++)

{

sum += (H\_matrix[i, k] \* code\_U[k, j]) % 2;

}

syndromes[i,j] = sum % 2;

}

}

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

{

for (int j = 0; j < 11; j++)

{

Console.Write(syndromes[i,j] + " ");

}

Console.WriteLine();

}

Console.WriteLine("\nКодирование");

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 11; j++)

{

sum = 0;

for (int k = 0; k < 4; k++)

{

sum += (bincodes[i, k] \* G\_matrix[k, j]) % 2;

}

code\_Matrix[i, j] = sum % 2;

}

}

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 11; j++)

{

Console.Write(code\_Matrix[i, j] + " ");

}

Console.WriteLine();

}

}

Вывод: В ходе в ходе выполнения лабораторной работы были приобретены навыки моделирования передачи сообщения по каналу с помехами.