**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

|  |
| --- |
| Определение и исправление ошибок в сообщении. Код Хемминга |

Руководитель А.Н. Бочаров

подпись, дата инициалы, фамилия

Обучающийся БПИ22-02, 221217039 Н.Ю. Сюткин

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2023 г.

# ЦЕЛЬ РАБОТЫ

Закрепление знаний по методам кодирования информации.

# порядок выполнения работы

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

# постановка задачи

Построить код Хемминга для исправления одиночной ошибки и обнаружения двойной ошибки.

Код должен предусматривать возможность посылки n сообщений.

1. Рассчитать параметры кода: nи , nк , n. Вариант задания определяется преподавателем.

2. Привести пример построения 10 кодовых сообщений.

3. Показать процедуру исправления ошибки в одной из позиций и обнаружения двойной ошибки.

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

Построение кода вручную.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | a5 | a6 | a7 |
| k1 | k2 | 0 | K3 | 0 | 1 | 1 |

a1+a3+a5+a7 = k1+0+0+1=0 => k1 = 1

a2+a3+a6+a7 = k2+0+1+1=0 => k2 = 0

a4+a5+a6+a7 = k3+0+1+1=0 => k3 = 0

Получаем код.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | a5 | a6 | a7 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 |

Модифицируем полученный код, добавлением еще одного разряда.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

Кол-во единиц нечетное => k4 = 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

Проверим полученный код на ошибки. Первым этапом проверим кол-во единиц. Кол-во единиц четное => либо в коде 2 ошибки, либо ошибок нет.

Выполним второй этап проверки:

a1+a3+a5+a7 = 1+0+0+1= 0

a2+a3+a6+a7 = 0+0+1+1= 0

a4+a5+a6+a7 = 0+0+1+1= 0

Вектор ошибок S = (0 0 0 0) => ошибок нет.

Допустим ошибку в 5-м разряде кода. Код с ошибкой.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

Проверим код на ошибки. Первым этапом проверим кол-во единиц. Кол-во единиц нечетное => значит в коде точно есть ошибка.

Выполним второй этап проверки:

a1+a3+a5+a7 = 1+0+1+1 = 1

a2+a3+a6+a7 = 0+0+1+1 = 0

a4+a5+a6+a7 = 0+1+1+1 = 1

Вектор ошибок S = (1 0 1) = 510 => таким образом в 5-м разряде была найдена ошибка.

Теперь допустим 2 ошибки в коде. Изначальный код

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

Код с ошибками во 3-м и в 4-м разряде.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

Проверим код на ошибки. Первым этапом проверим кол-во единиц. Кол-во единиц четное => значит в коде есть двойная ошибка, либо ошибок нет.

Выполним второй этап проверки:

a1+a3+a5+a7 = 1+1+0+1= 1

a2+a3+a6+a7 = 0+1+1+1= 1

a4+a5+a6+a7 = 1+0+1+1= 1

Вектор ошибок S = (1 1 1) ненулевой, следовательно обнаружена двойная ошибка.

Таким образом, в коде было обнаружено 2 ошибки.

Далее была разработана программа, которая строит модифицированные коды Хемминга и выполняет декодирование

Программу представляет ряд методов и консольное меню.

Метод HemmingCoding реализует создание расширенного кода Хемминга:

public static int[] HemmingCoding(int[] code, int n\_u, int n\_k, int n)

{

int[] ai = new int[n + 1];

ai[0] = 999;

ai[1] = 999;

ai[2] = code[0];

ai[3] = 999;

ai[4] = code[1];

ai[5] = code[2];

ai[6] = code[3];

ai[0] = (ai[2] + ai[4] + ai[6]) % 2;

ai[1] = (ai[2] + ai[5] + ai[6]) % 2;

ai[3] = (ai[4] + ai[5] + ai[6]) % 2;

ai[7] = ai.Sum() % 2;

return ai;

}

Метод FixMistake представляет инструмент для обнаружения двойной и исправления одиночной ошибки:

public static void FixMistake(int[] codes)

{

int sum = 0;

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

{

sum += codes[i];

}

if(sum % 2 == 0)

{

Console.WriteLine("\nСумма единиц ЧЕТНАЯ --> в коде двойная ошибка/ошибки нет");

int s1 = (codes[0] + codes[2] + codes[4] + codes[6]) % 2;

int s2= (codes[1] + codes[2] + codes[5] + codes[6]) % 2;

int s3 = (codes[3] + codes[4] + codes[5] + codes[6]) % 2;

if ((s1 + s2 + s3) > 0)

Console.WriteLine("Синдром НЕнулевой --> ошибка двойная");

else

Console.WriteLine("Синдром нулевой --> ошибки нет");

}

else

{

Console.WriteLine("\nСумма единиц НЕЧЕТНАЯ --> в коде одиночная ошибка");

int s1 = (codes[0] + codes[2] + codes[4] + codes[6]) % 2;

int s2 = (codes[1] + codes[2] + codes[5] + codes[6]) % 2;

int s3 = (codes[3] + codes[4] + codes[5] + codes[6]) % 2;

string number = s3.ToString() + "," + s2.ToString() + "," + s1.ToString();

string number1 = s3.ToString() + s2.ToString() + s1.ToString();

Console.WriteLine("Синдром S: (" + number + ")");

int \_int = Convert.ToInt32(number1,2);

Console.WriteLine("Одиночная ошибка найдена --> находится в " + \_int + " разряде");

}

}

Метод RandomValues генерирует случайные 10 кодов. Так как вариант предусматривает 16 возможных вариантов, они были занесены в отдельный массив. Далее случайным образом по очереди выбираются 10 индексов из данного массива и коды под этими индексами используются в программе. После выбора элемента, он сразу удаляется из массива с 16 вариантами. (Например из 16 был случайно выбран 1 => стало 15 вариантов для случайного выбора и так далее). Сделано это для оптимизации процесса, так как вероятность выбора случайных 10 кодов из 16 абсолютно случайным образом/подбором невысока.

В программе используется метод DeleteElem, реализованный для удаления элемента под соответствующим индексом в заданном массиве.

public static void RandomValues(int[][] codes)

{

Random rnd = new Random();

int[][] array = new int[16][];

array[0] = new int[] {0, 0, 0, 0 };

array[1] = new int[] { 0, 0, 0, 1 };

array[2] = new int[] { 0, 0, 1, 0 };

array[3] = new int[] { 0, 0, 1, 1 };

array[4] = new int[] { 0, 1, 0, 0 };

array[5] = new int[] { 0, 1, 0, 1 };

array[6] = new int[] { 0, 1, 1, 0 };

array[7] = new int[] { 0, 1, 1, 1 };

array[8] = new int[] { 1, 0, 0, 0 };

array[9] = new int[] { 1, 0, 0, 1 };

array[10] = new int[] { 1, 0, 1, 0 };

array[11] = new int[] { 1, 0, 1, 1 };

array[12] = new int[] { 1, 1, 0, 0 };

array[13] = new int[] { 1, 1, 0, 1 };

array[14] = new int[] { 1, 1, 1, 0 };

array[15] = new int[] { 1, 1, 1, 1 };

int index = 0;

int count = 16;

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

{

index = rnd.Next(0, count);

codes[i] = array[index];

count--;

array = DeleteElem(array, index);

}

}

Основной метод Main, реализующий консольное меню.

static void Main()

{

//Инициализируем массивы с кодами

int[][] codes\_ = new int[0][];

int ans = 0, ans2 = 0;

Console.WriteLine("Выберите пункт меню: ");

do

{

Console.WriteLine("1 - Кодирование");

Console.WriteLine("2 - Декодирование");

Console.WriteLine("3 - Выход из программы");

ans = int.Parse(Console.ReadLine());

switch (ans)

{

case 1:

codes\_ = new int[10][];

for (int i = 0; i < codes\_.Length; i++)

{

codes\_[i] = new int[4];

}

//Генерируем случайную информационную часть

RandomValues(codes\_);

//Считаем разряды

int n\_u = 4;

int n\_k = (int)Round(Log((n\_u + 1) + Log(n\_u + 1, 2), 2));

int n = n\_u + n\_k;

Console.WriteLine("Разряды: " +

"\nИнформационные n(и) - " + n\_u +

"\nКонтрольные n(к) - " + n\_k +

"\nОбщая разрядность кода n - " + n);

for (int i = 0; i < codes\_.Length; i++)

{

Console.Write((i + 1) + ")");

codes\_[i] = HemmingCoding(codes\_[i], n\_u, n\_k, n);

for (int j = 0; j < codes\_[i].Length; j++)

{

Console.Write(codes\_[i][j]);

}

Console.WriteLine();

}

break;

case 2:

if (codes\_.Length <= 0)

Console.WriteLine("Сгенерируйте коды (Пункт 1)");

else

{

do

{

Console.WriteLine("1 - Ввести и исправить одиночную ошибку");

Console.WriteLine("2 - Ввести и определить двойную ошибку");

Console.WriteLine("3 - Вернуться в главное меню");

ans2 = int.Parse(Console.ReadLine());

switch (ans2)

{

case 1:

int[][] temp\_codes = codes\_;

for (int i = 0; i < temp\_codes.Length; i++)

{

Console.Write((i + 1) + ")");

for (int j = 0; j < temp\_codes[i].Length; j++)

{

Console.Write(temp\_codes[i][j]);

}

Console.WriteLine();

}

Console.Write("Выберите код для одиночной ошибки: ");

int index\_code = int.Parse(Console.ReadLine()) - 1;

Console.Write("Введите индекс, где совершить ошибку: ");

int index\_mistake = int.Parse(Console.ReadLine()) - 1;

if (temp\_codes[index\_code][index\_mistake] == 0)

temp\_codes[index\_code][index\_mistake] = 1;

else temp\_codes[index\_code][index\_mistake] = 0;

for(int i = 0; i < codes\_[index\_code].Length; i++)

{

Console.Write(codes\_[index\_code][i]);

}

Console.Write(" --> ");

for(int i = 0; i < temp\_codes[index\_code].Length; i++)

{

if (i == index\_mistake)

Console.Write("(" + temp\_codes[index\_code][i] + ")");

else

Console.Write(temp\_codes[index\_code][i]);

}

FixMistake(temp\_codes[index\_code]);

break;

case 2:

temp\_codes = codes\_;

for (int i = 0; i < temp\_codes.Length; i++)

{

Console.Write((i + 1) + ")");

for (int j = 0; j < temp\_codes[i].Length; j++)

{

Console.Write(temp\_codes[i][j]);

}

Console.WriteLine();

}

Console.Write("Выберите код для двойной ошибки: ");

index\_code = int.Parse(Console.ReadLine());

Console.WriteLine("Введите два индекса для совершения ошибки. ");

Console.Write("Первый: ");

int index\_mistake1 = int.Parse(Console.ReadLine()) - 1;

Console.Write("Второй: ");

int index\_mistake2 = int.Parse(Console.ReadLine()) - 1;

if (temp\_codes[index\_code][index\_mistake1] == 0)

temp\_codes[index\_code][index\_mistake1] = 1;

else temp\_codes[index\_code][index\_mistake1] = 0;

if (temp\_codes[index\_code][index\_mistake2] == 0)

temp\_codes[index\_code][index\_mistake2] = 1;

else temp\_codes[index\_code][index\_mistake2] = 0;

for (int i = 0; i < codes\_[index\_code].Length; i++)

{

Console.Write(codes\_[index\_code][i]);

}

Console.Write(" --> ");

for (int i = 0; i < temp\_codes[index\_code].Length; i++)

{

if (i == index\_mistake1 || i == index\_mistake2)

Console.Write("(" + temp\_codes[index\_code][i] + ")");

else

Console.Write(temp\_codes[index\_code][i]);

}

FixMistake(temp\_codes[index\_code]);

break;

}

} while (ans2 != 3);

} break;

}

} while (ans != 3);

}

}

Демонстрация начального меню программы (рисунок 1).

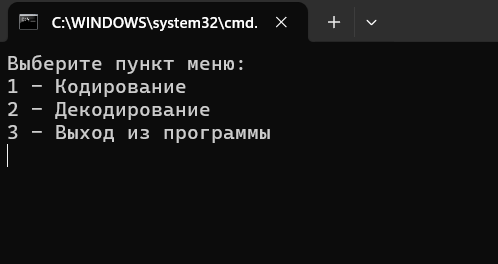


Рисунок 1 – Начальное меню программы

Путем выбора первого пункта меню, мы можем сгенерировать нужное нам кол-во расширенных кодов Хемминга (рисунок 2).

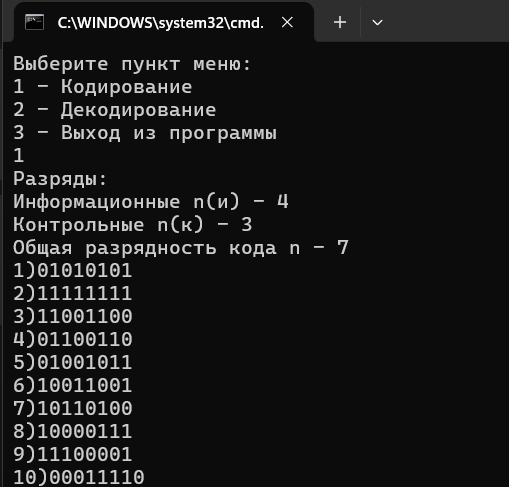


Рисунок 2 – Расширенные коды Хемминга

Выбрав второй пункт меню, мы можем перейти к подменю декодирования (рисунок 3).

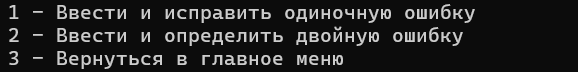


Рисунок 3 – Подменю декодирования

Выбрав первый пункт подменю, можно искусственно задать одиночную ошибку, обнаружить и исправить её (Рисунок 4).

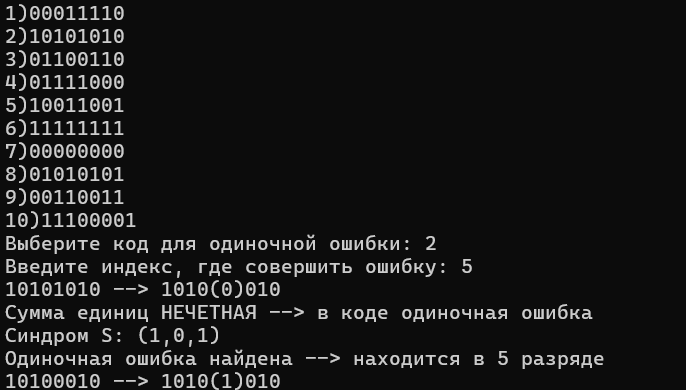


Рисунок 4 – Обнаружение и исправление одиночной ошибки

Выбрав второй пункт меню, можно искусственно задать двойную ошибку и обнаружить её.

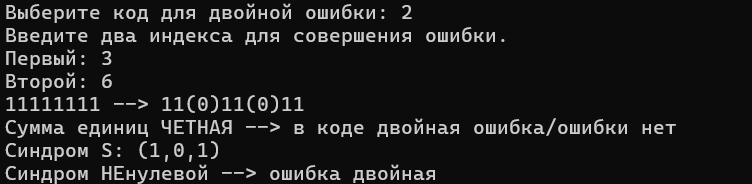


Рисунок 5 – Обнаружение двойной ошибки

# ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое корректирующий код?

Корректирующие или помехоустойчивые коды — это коды, позволяющие

обнаруживать и исправлять ошибки, происходящие в процессе передачи из-за

влияния помех.

2. Как обнаруживаются ошибки при передаче информации с помощью корректирующего кода?

Возможность обнаружения ошибок состоит в том, что для передачи используют не все N0 = 2n возможных кодовых комбинаций, а лишь часть их N < N0. Используемые комбинации N называются разрешенными, а остальные комбинации N0 – N – запрещенными.

3. Как рассчитать параметры корректирующего кода?

4. Как определить кодовое расстояние?

Для того чтобы определить кодовое расстояние, достаточно просуммировать (по модулю два) кодовые комбинации и подсчитать число единиц в полученной комбинации.

5. Как определить минимальное кодовое расстояние?

Необходимое для обнаружения и исправления ошибок минимальное кодовое расстояние вычисляется по следующему выражению:

r – число обнаруживаемых ошибок,

s – исправляемых ошибок.

7. Как строится код Хемминга.

1) Вычисление основных параметров (nк, nи, n).

2) Построение кода с корректирующими разрядами.

3) Вычисление корректирующих разрядов.

4) Заполнение соответствующих вычисленных разрядов.

8. Как проводится процедура декодирования кода Хемминга?

1) Вычисление синдрома (S)

2) Преобразование его обратного вида в десятичную систему счисление из двоичной.

3) Замена соответствующего вычисленному значению номера в коде на противоположный.

9. Что такое пропускная способность канала связи?

Пропускная способность канала связи это максимальная скорость передачи данных через канал.

С = Vx\*= Vx [1+p+(1-p)

Vx – средняя скорость передачи символов

– максимальное кол-во информации на 1 символ

10. В чем суть теоремы Шеннона для дискретного канала с помехами?

Если источник информации имеет энтропию H(z), а канал связи обладает пропускной способностью, то отсюда следует:

1) Сообщение, вырабатываемое источником, всегда можно закодировать так, чтобы скорость Vz их передачи была сколь угодно близка к значению Vzmax=c/(H(z))

2) Не существует способа кодирования, позволяющего вести передачу со скоростью выше Vzmax и с малой вероятностью ошибки.

3)

11. Как описать двоичный канал?

12. Как определить пропускную способность двоичного канала?

С = Vx\*

Vx – средняя скорость передачи символов

13. Что представляет собой вектор ошибок при обнаружении и исправлении ошибок в сообщении?

Вектор ошибок S = (0 1 0 1) = 510 => таким образом в 5-м разряде была найдена ошибка.

# ВЫВОДЫ

В ходе выполнения лабораторной работы были получены навыки создания кода Хемминга и расширенной версии. Также освоено обнаружение одиночной и двойной ошибки, и исправление одиночной ошибки.